Design Computing and Cognition '06

Edited by
JOHN S. GERO

Springer

DESIGN COMPUTING AND COGNITION ’06

Design Computing and Cognition '06

Edited by

JOHN S. GERO
University of Sydney, NSW, Australia

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN-10 1-4020-5130-1 (HB)
ISBN-13 978-1-4020-5130-2 (HB)
ISBN-10 1-4020-5131-X (e-book)
ISBN-13 978-1-4020-5131-9 (e-book)

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

www.springer.com

Printed on acid-free paper

All Rights Reserved
© 2006 Springer
No part of this work may be reproduced, stored in a retrieval system, or transmitted
in any form or by any means, electronic, mechanical, photocopying, microfilming, recording
or otherwise, without written permission from the Publisher, with the exception
of any material supplied specifically for the purpose of being entered
and executed on a computer system, for exclusive use by the purchaser of the work.

Printed in the Netherlands.

TABLE OF CONTENTS

Preface **ix**

List of References **xi**

REPRESENTATION IN DESIGN **1**

Representing style by feature space archetypes 3
Sean Hanna

A digital bridge for performance-based design 23
Dirk A Schwede

Building connectivity models in design: Representations and tools to support cognitive preferences 41
René Keller, Claudia M Eckert and P John Clarkson

Geometric, cognitive and behavioral modeling of environmental users: Integrating an agent-based model and a statistical model into a user model 61
Wei Yan and Yehuda E Kalay

EARLY STAGES OF DESIGN **81**

Exploration through drawings in the conceptual stage of product design 83
Miquel Prats and Chris F Earl

Digital sketching in a multi-actor environment 103
Alexander Koutamanis

The Designosaur and the furniture factory 123
Yeonjoo Oh, Gabriel Johnson, Mark Gross and Ellen Yi-Luen Do

Industrial mechanical design: the “ids” case study 141
Stefania Bandini and Fabio Sartori

DESIGN METHODOLOGIES **161**

System development methodologies: A knowledge perspective 163
Warren Kerley and Tony Holden

Analogical matching using device-centric and environment-centric representations of function 183
Greg P Milette and David C Brown

Design operators to support organisational design 203
Catholijn M Jonker, Alexei Sharpanskykh, Jan Treur and Pinar Yolum

Bayesian networks for design 223
Peter Matthews

COGNITIVE STUDIES OF DESIGNERS **243**

A comparative study of problem framing in multiple settings 245
Thomas Kvan and Song Gao

Comparing entropy measures of idea links in design protocols 265
Jeff W T Kan, Zafer Bilda and John S Gero

Analysing the Emotive Effectiveness of rendering styles 285
Raji Tenneti and Alex Duffy

Impact of collaborative virtual environments on design behaviour 305
Mary Lou Maher, Zafer Bilda and Leman Figen Gül

DESIGN THINKING **323**

Content-based analysis of modes in design engineering 325
Pertti Saariluoma, Kalevi Nevala and Mikko Karvinen

Buildings and affordances 345
Alexander Koutamanis

The role of preconceptions in design: Some implications for the development of computational design tools 365
Patrick H T Janssen

How am I doing? The language of appraisal in design 385
Andy Dong

FORMAL METHODS IN DESIGN **405**

A function-behaviour-structure ontology of processes 407
John S Gero and Udo Kannengiesser

From form to function: From SBF to DSSBF 423
Patrick W Yaner and Ashok K Goel

Formal description of concept-synthesizing process for creative design 443
Yukari Nagai and Toshiharu Taura

Robustness in conceptual designing: Formal criteria 461
Kenneth A Shelton and Tomasz Arciszewski

GRAMMARS IN DESIGN **481**

An urban grammar for the Medina of Marrakech 483
José P Duarte, Gonçalo Ducla-Soares, Luisa G Caldas and João Rocha

CAD Grammars: Combining CAD and Automated Spatial Design 503
Peter Deak, Glenn Rowe and Chris Reed

Combining evolutionary algorithms and shape grammars to generate branded product design 521
Mei Choo Ang, Hau Hing Chau, Alison McKay and Alan de Pennington

A semantic validation scheme for graph-based engineering design grammars 541
Stephan Rudolph

LEARNING IN DESIGN **561**

Inductive machine learning of microstructures 563
Sean Hanna and Siavash Haroun Mahdavi

Learning from 'superstar' designers 583
Paul A Rodgers

The improvement of design solutions by means of a question-answering technique 603
Constance Winkelmann and Winfried Hacker

Contextual cueing and verbal stimuli in design idea generation 619
Lassi A Liikkanen and Matti K Perttula

DESIGN COLLABORATION **633**

Communicating, integrating and improving multidisciplinary design narratives 635
John Haymaker

Enhanced design checking involving constraints, collaboration, and assumptions 655
Janet Burge, Valerie Cross, James Kiper, Pedrito Maynard-Zhang and Stephan Cornford

From architectural sketches to feasible structural systems 675
Rodrigo Mora, Hugues Rivard, Roland Juchmes, and Pierre Leclercq

DesignWorld: A multidisciplinary collaborative design environment using agents in a virtual world 695
Michael Rosenman, Kathryn Merrick, Mary Lou Maher and David Marchant

Contact Authors' Email Addresses **711**

Author Index **713**

Preface

There is an increasing realization that design is part of the wealth creation of a nation and needs to be better understood. The continuing globalization of trade and the means of production has required nations to re-examine where their core contributions lie if is not in production efficiency. Design is a precursor to manufacturing for physical objects and is the precursor to implementation for virtual objects. As the value of designing increases, so the value of design research increases and as a consequence design research has moved from being an arcane field closer to centre stage.

There are now three sources for design research: design computing, design cognition and advanced technology. The foundations for much of design computing remains artificial intelligence with its focus on ways of representation and on processes that support simulation and generation. Artificial intelligence continues to provide an environmentally rich paradigm within which design research based on computational constructions can be carried out. Design cognition is founded on concepts from cognitive science, an even newer area than artificial intelligence. It provides tools and methods to study human designers in both laboratory and practice settings. It is beginning to allow us to test the claims being made about the effects of the introduction of advanced technologies into the design processes in addition to helping us understand the act of designing itself. Advanced technologies specifically to aid designing and those drawn into the design process from elsewhere provide a fertile area for researchers. All these areas are represented in this conference.

This conference series aims at providing a bridge between the two fields of design computing and design cognition. The confluence of these two fields is likely to provide the foundation for further advances in each of them.

The papers in this volume are from the *Second International Conference on Design Computing and Cognition (DCC'06)* held at the Technical University of Eindhoven, the Netherlands. They represent the state-of-the-art of research and development in design computing and design cognition. They are of particular interest to researchers, developers and users of

advanced computation in design and those who need to gain a better understanding of designing.

In these proceedings the papers are grouped under the following nine headings, describing both advances in theory and application and demonstrating the depth and breadth of design computing and of design cognition:

Representation in Design
Early Stages of Design
Design Methodologies
Cognitive Studies of Designers
Design Thinking
Formal Methods in Design
Grammars in Design
Learning in Design
Design Collaboration

There were over 100 submissions to the conference. Each paper was extensively reviewed by three referees drawn from the international panel of seventy-three active referees listed. The reviewers' recommendations were then assessed before the final decision on each paper was taken. Thanks go to them, for the quality of these papers depends on their efforts.

Mercèdes Paulini worked well beyond the call of duty to get these proceedings together and out in time. She deserves particular thanks for it was she who took the authors' final submissions and turned them into a uniform whole. The final quality of the manuscript is largely a result of her efforts and bears her mark.

John S Gero
University of Sydney
April 2006

LIST OF REFEREES

Henri Achten, Eindhoven University of Technology, The Netherlands
Omer Akin, Carnegie Mellon University, USA
Tom Arciszewski, George Mason University, USA
Uday Athavankar, Indian Institute of Technology, India
Jean-Paul Barthès, Université de Compiègne, France
Kirsty Beilharz, University of Sydney, Australia
Peter Bentley, University College London, UK
Bo-Christer Björk, Royal Institute of Technology, Sweden
Nathalie Bonnardel, University of Provence, France
Dave Brown, Worcester Polytechnic Institute, USA
Jon Cagan, Carnegie Mellon University, USA
Scott Chase, University of Strathclyde, UK
Maolin Chiu, National Cheng Kung University, Taiwan
John Clarkson, University of Cambridge, UK
Sambit Datta, Deakin University, Australia
Bauke de Vries, Eindhoven University of Technology, The Netherlands
Ellen Yi-Luen Do, University of Washington, USA
Andy Dong, University of Sydney, Australia
Jose Duarte, Instituto Superior Technico, Portugal
Alex Duffy, University of Strathclyde, UK
Chris Earl, The Open University, UK
Claudia Eckert, University of Cambridge, UK
Christian Freksa, Universitaet Bremen, Germany
Haruyuki Fujii, Tokyo Institute of Technology, Japan
John Gero, University of Sydney, Australia
Alberto Giretti, IDAU (Universita di Ancona), Italy
Ashok Goel, Georgia Institute of Technology, USA
Gabi Goldschmidt, Technion University of Technology, Israel
Mark Gross, University of Washington, USA
David Gunaratnam, University of Sydney, Australia
John Haymaker, CIFE, Stanford University, USA
Ann Heylighen, Arenbergkasteel, Belgium
Koichi Hori, University of Tokyo, Japan
Ludger Hovestadt, Federal Institute of Technology Zurich, Switzerland
TaySheng Jeng, National Cheng Kung University, Taiwan
Richard Junge, Technical University Munich, Germany
Julie Jupp, University of Cambridge, UK
Udo Kannengiesser, University of Sydney, Australia
Ruediger Klein, DaimlerChrysler, Germany
Terry Knight, MIT, USA
Ramesh Krishnamurti, Carnegie Mellon University, USA
Tom Kvan, University of Sydney, Australia
Bryan Lawson, University of Sheffield, UK
Pierre Leclercq, University of Liége, Belgium
Andrew Li, Chinese University of Hong Kong, Hong Kong
Mary Lou Maher, University of Sydney, Australia
Kumiyo Nakakoji, University of Tokyo, Japan
Barry O'Sullivan, University of College Cork, Ireland
Rivka Oxman, Technion Israel Institute of Technology, Israel
Jens Pohl, California Polytechnic State University, USA
Rabee Reffat, College of Environmental Design, Saudi Arabia
Yoram Reich, Tel Aviv University, Israel
Hugues Rivard, ETS - University of Quebec, Canada
Michael Rosenman, University of Sydney, Australia
Rob Saunders, University of Sydney, Australia
Gerhard Schmitt, Swiss Federal Institute of Technology Zurich, Switzerland
Kristi Shea, University of Cambridge, UK
Greg Smith, University of Sydney, Australia

LIST OF REFERENCES

Ian Smith, Federal Institute of Technology (EPFL), Switzerland
Tim Smithers, VICOMTech, Spain
Ricardo Sosa, Instituto Technologico y de Estudios Superiores de Monterrey, Mexico
Ram Sriram, NIST, USA
George Stiny, MIT, USA
Rudi Stouffs, University of Technology Delft, The Netherlands
Masaki Suwa, Chukyo University, Japan
Ming-Xi Tang, Hong Kong Polytechnic University, Italy
Hsien Hui Tang, College of Management, Chang Gung University, Taiwan
Bige Tuncer, Delft University of Technology, The Netherlands
Barbara Tversky, Stanford University, USA USA
Patrick Yaner, Georgia Institute of Technology,
Andrew Vande Moere, University of Sydney, Australia

REPRESENTATION IN DESIGN

Representing style by feature space archetypes
Sean Hanna

A digital bridge for performance-based design
Dirk A Schwede

Building connectivity models in design: Representations and tools to support cognitive preferences
René Keller, Claudia Eckert and P John Clarkson

Geometric, cognitive and behavioral modeling of environmental users: Integrating an agent-based model and a statistical model into a user model
Wei Yan and Yehuda Kalay

REPRESENTING STYLE BY FEATURE SPACE ARCHETYPES

Description and Emulation of Spatial Styles in an Architectural Context

SEAN HANNA
University College London, UK

Abstract. Style is a broad term that could potentially refer to any features of a work, as well as a fluid concept that is subject to change and disagreement. A similarly flexible method of representing style is proposed based on the idea of an archetype, to which real designs can be compared, and tested with examples of architectural plans. Unlike a fixed, symbolic representation, both the measurements of features that define a style and the selection of those features themselves can be performed by the machine, making it able to generalise a definition automatically from a set of examples.

1. Introduction

At its core, style is what distinguishes one group of works from another. This paper proposes that we can define a style using an *archetype*, an ideal model comprised of the features that exemplify the style. This concept differs from the description of a type, or category into which particular examples can fall, and from that of a prototype, precedent or case, which are actual instances on which later examples can be modelled. An archetype is something between the two, a generalisation that can not exist materially, yet matches and is compared to many actual instances. This is almost certainly not a real example, but an abstraction made up of only those features necessary to differentiate it from other archetypes.

Many approaches to style are based on explicit symbolic representations (where fixed concepts are mapped to named variables) or rule systems. These can tell us useful things about designs and how they can be made, but are inflexible. They reveal some of the ways we learn about styles pedagogically, but typically fixed, historical ones. By contrast this work proposes a method to automatically derive representations from real examples of design.

It is based on the mapping of design examples in a high dimensional feature space, and uses methods of dimensionality reduction of this space to

J.S. Gero (ed.), Design Computing and Cognition '06, 3–22.
© 2006 *Springer. Printed in the Netherlands.*

yield an archetype that describes the style. This can be used to classify, and as a measure to generate new designs. The use of a feature space agrees with our own intuitive ability to evaluate designs as being stylistically nearer or farther from one another, and is commonly applied in machine learning, in which a space is constructed in which each dimension is a measurement of a particular feature, and so each example can be represented as a single point. The nearest neighbour algorithm (e.g. Duda et al. 2001), for instance, classifies an unknown example of data by simply measuring its distance to previously known and labelled examples, or prototypes.

Two innovations are proposed over such existing methods. The first is that the archetype is a generalisation that combines both the concept of the ideal example and the particular space in which it is measured. In the nearest neighbour algorithm, a prototype is a real example of data, and all examples are measured within the same space. The archetypes presented here are measured in a lower dimensional space consisting only of the features relevant to that style, and each archetype may be measured in a different feature space. The archetype, then, is a point in a feature space consisting of dimensions in which examples of a particular style are closely clustered, and examples of other styles are distant. It is comprised of both point and space.

This provides a method for analysis of existing designs, but not synthesis of new ones. Rule-based definitions can be useful because they can be followed to produce new designs, whereas a classification algorithm by itself clearly cannot. The second innovation incorporates the notion of affordances (Gibson 1979), to consider the design process as a continual evaluation and choice between possible alternatives as the development of the design progresses. These choices can be made by repeated measurement against the ideal that the archetype represents.

The approach was tested within the design domain of architecture, using the plan in particular. This paper implements the two major aspects of the approach in Sections 3 and 4. The first deals with analysis, and begins by providing methods by which plans of real buildings can be embedded in a feature space such that those that are similar fall near to one another. This yields a way in which all styles might be understood and represented by a computer, which is not based on any predefined symbolic representation. The second part refines this spatial embedding, and combines a very simple generative process to synthesise designs. An archetype is defined from a set of examples and used to guide new designs in the same style.

2. Related Techniques

Several techniques from related fields of machine vision and space syntax are relevant to this work. They are outlined here along with a discussion of various existing approaches to style.

2.1. APPROACHES TO ARCHITECTURAL STYLE

The need to differentiate is fundamental to communication. Gombrich (1960) suggests that art provides categories by which to sort our impressions: 'without some starting point, some initial schema, we could never get hold of the flux of experience.' Like a game of 'Twenty Questions', where the actual identity of a concept is gradually communicated through trial and error, the symbols of art do not represent reality in all its detail, but only what is necessary to convey the message. In his view the set of symbols used constitute the style, and one general approach to representing style is rooted in identifying the equivalent of these symbols, either as features or generative rules of the work.

Architectural style has been represented a number of ways, both for analysis of existing designs, and for synthesis of new ones. Analytical methods have been proposed that model similarity or group designs based on a count of pre-defined features. Chan (1994) uses a list of material and compositional features to evaluate buildings as to their style, and finds a correlation with the judgements of test subjects. Experiential qualities of architecture have also been structured or mapped to rule sets to guide architects (Alexander et al. 1977; Koile 2004 1997), and this approach has been implemented by Koile (1997) in an expert system that is also able to recognise style as a count of defined design characteristics.

Rule systems have also been developed to generate new designs in a particular style, such as Hersey and Freedman's (1992) computer implementation to create possible Palladian villas. Shape grammars (Stiny 1976, 1980) are possibly the most widely studied, and have been successful. They have yielded examples in the apparent styles of Palladian villas (Stiny and Mitchell 1978) and Frank Lloyd Wright's prairie houses (Koning and Eizenberg 1981). Recent approaches have expanded the method to allow decompositions on finished or existing designs to generate new rules for design exploration (Prats et al. 2006).

As an approach to style, a style is often (e.g. the examples above) encoded with a specific grammar, unlike linguistic grammars that generate a particular language with any number of possible styles. A creative human then works with the shape grammar to make a specific design within the style. As a tool for analysis, the grammar or rule set is constructed by a human designer, a fully automatic process seen as undesirable or impossible (Knight 1998). In its generative capacity it is then followed by a user choosing which rule to apply at each stage in the process to create designs comparable to originals.

Another approach to style proposes that it is not defined by clear and predetermined features or rules, but can be quantified by various measurements taken from examples of the works. More general analytical

techniques using information theoretic measures have been used to measure distance between individual plans (Jupp and Gero 2003), and to derive values for measurements such as entropy (Gero and Kazakov 2001), morphology and topology (Jupp and Gero 2006) that can be used to evaluate examples of a style. These have the advantage of quantifying designs of any style as real values on the same scales, so that variations within or between styles can be measured uniformly.

It is this second approach that is preferred in the present work, for several reasons. While the setting of explicit rules or feature definitions can tell us interesting things about a style, they are often a simplification that will either produce some designs that would be considered outside the style, or fail to produce all possibilities within (Hersey and Freedman 1992, ch. 2). But there may be no particular class of measures that we can specify in advance to contain the description of all styles. While style is often considered (as in Gombrich 1960) the method of expression as opposed to the content, Goodman (1975) argues for a broader definition of style to include aspects of both *what* is expressed in addition to *how*. In his definition, the style 'consists of those features of the symbolic functioning of a work that are characteristic of author, period, place or school', and these can only be determined in relation to the works, not beforehand.

This present work differs from previous approaches in that design examples will be evaluated by a general analysis, then the relevant features determined automatically in the definition of the archetype. By so doing, both the processes of defining a style and building examples of it can be performed by a machine.

2.2. FEATURE DESCRIPTION BY DIMENSIONALITY REDUCTION IN OTHER FIELDS

To automatically generalise a description of a style, either as a set of relevant features or a generative rule, from a set of given examples is more difficult than setting it in advance, but this approach is beginning to be explored in other stylistic domains, such as musical style (Tillmann et al. 2004). It is based on more firmly established techniques of machine classification and learning in other fields, particularly machine vision, as used in applications such as face recognition and robot navigation.

Dimensionality reduction is often used in applications from face recognition to linguistic analysis to infer distinguishing features from a given set of high-dimensional data. Principal component analysis (PCA) provides a new set of axes aligned to the characteristic vectors, or eigenvectors of the covariance matrix of the data set. The principal components of face images, for example, referred to as 'eigenfaces' (Turk and Pentland 1991), are used

by face recognition software to effectively quantify a new face by how it measures against each, and its best match found from existing data.

More closely related to our experience of architecture is the problem of a robot visually navigating through a real environment. Rather than explicit labelling, it has been found preferable to allow the computer to come up with its own concepts: an unsupervised learning of the visual features of the environment. In work on robot navigation of unstructured environments, Durrant-White (2004) has used dimensionality reduction on the image data recorded by the moving camera of a robot's visual system.

2.3. REPRESENTATION OF SPATIAL FEATURES AS GRAPHS

To represent stylistic features of space, the computer requires an appropriate substitute for the sense data provided by the images or sound digitisations in the applications above – a method through which to perceive experiential qualities of the building. Two related space syntax techniques both provide an approximation of how people actually move through or use a space, using only an analysis of the plan. *Visibility graph* analysis quantifies the connectivity of a set grid of points within a space by the unobstructed sightlines that exist between them. From these, various measures such as integration, connectivity or mean depth of points can be used to derive a statistical analysis of the space based on the plan. (Turner et al. 2001) *Axial graph* analysis quantifies the connectivity of sightlines themselves, derived from plan vertices (Hillier et al. 1983; Hillier and Hanson 1984).

Properties of visibility and axial graphs have been shown to be strongly related to both spatial perception and resulting behaviour of people within spaces. Strong correlations have been found with measures of visibility graphs and observed way-finding, movement and use in buildings (Turner et al. 2001), and urban pedestrian movement (Desyllas and Duxbury 2001). Axial graphs have likewise been shown to be closely related to directly observed movement (Peponis et al. 1989; Hillier et al. 1993), building use and social interaction (Spiliopoulou and Penn 1999), and indirect behaviour such as land values and crime (Desyllas 2000; Hillier and Shu 2001).

Of the two methods, axial graphs are used in this work as the sightline endpoints are not predetermined and therefore invariant to plan scaling or rotation. The details of the algorithm are beyond the scope of this paper (see Turner 2005), but measurements such as connectivity and integration taken from the graph in axial analysis quantify the kinds of experience of that space as inhabited by a large number of bodies. Rather than these pre-determined features, the raw graph itself can be easily taken as the machine's generic experiential input.

2.3.1. Measuring between graphs

Several approaches to similarity measurement have been based on small graphs of adjacency or connectivity of spaces in plan. Dalton and Kirsan (2005) use the number of transformations necessary to derive one such graph from another to measure the similarity between buildings, and Jupp and Gero (2003) suggest an analysis based on similarity and complexity measures of semantic graphs. With larger and more complex graphs as generated by axial lines, calculation of similarity becomes more difficult, but this can be overcome with graph spectral analysis. Spectral analysis of a graph uses the eigenvalues and eigenvectors of its connectivity matrix, and is relevant to the kinds of analysis considered here.

3. Analysis: Representation of a Style in Feature Space

The archetype feature space must be derived initially from a general input capable of containing features relevant to all styles. This section tests both the acceptability of the axial graph as sense input, and the general use of a feature space for real building plans.

It is proposed that the description of a style is inherent in the examples of that style, and so examples of one style are objectively more similar to one another than to examples of other styles. The stylistic description should therefore not be needed a priori to make this distinction, but the algorithm should be able to make the same kinds of classifications that we would, without explicit training. In this section a feature space is found for a group of buildings by unsupervised dimensionality reduction as in the examples in Section 2.2. The examples are found to naturally fall into recognisable clusters within the space without the need for explicit labelling, and these correspond to the buildings' inherent similarities. This will provide the basis from which to derive an archetype from the examples.

3.1. DESCRIPTION OF PLAN STYLES IN A FEATURE SPACE

An initial test to describe styles in a feature space used a set of 24 plans, taken from various twentieth century iconic buildings (Weston 2004). This involved a simple dimensionality reduction of a feature space to confirm the hypothesis that proximity does indeed indicate stylistic similarity.

Axial graphs were constructed for each of the 24 samples, and this data – in effect a set of binary adjacency matrices – was taken as the computer's raw sense data, or experience of each of the spaces. Analysis was performed using Depthmap software, which constructed a minimal axial graph based entirely on plan input and an objective algorithm (Turner et al. 2001). Figure 1 displays the lines of sight in Frank Lloyd Wright's Fallingwater shaded to indicate the degree of spatial integration. Darker lines clearly reveal the higher traffic zones that link the rooms of the house. The spectrum of this

graph was taken to form the raw feature vector so as to deal directly with the connectivity in the axial line graph. The spectra were created by ordering the eigenvalues of these adjacency matrices in order of decreasing magnitude, thus yielding a feature vector for each plan.

With the spatial analysis of each plan quantified as a feature vector, the example buildings can then be plotted in a high dimensional feature space, with each value in the spectrum on a different dimensional axis. PCA determines, for a given set of data, the combination of dimensions in which it is most likely to vary, and these are used as new axes in a reduced version of the space that captures the essential statistical features of the data set. A reduction of the plans' feature space based on the first two principal components of the set is shown in Figure 2. The dimensions of this new feature space are strictly computational, and are meaningful only in a statistical sense, rather than in the sense that they could be easily described. The first component, on the horizontal axis, represents a combination of the features in which the plans are judged by the *algorithm* to differ the most.

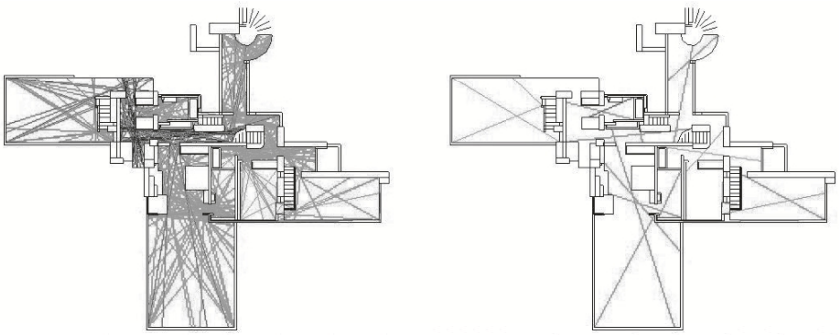


Figure 1. A complete axial graph of Frank Lloyd Wright's Fallingwater (*left*) and the reduced minimal graph (*right*).

Yet it can be seen that these include meaningful features such as typology (most houses toward the right and larger, public buildings to the left) as well as general stylistic trends. The shaded groups indicate the proximity of most of the axially symmetrical, pre-Modernist buildings to one another, as well as rough zones of early, and of high modernist buildings, typically with spaces that flow into one another articulated by shifting orthogonal planes. There is an approximate chronological order from left to right, seen clearly in the buildings by Wright, the Villa Savoye and contempory Maison de Verre are next to one another, and van der Rohe's two works are virtually overlapping.

The divisions in the diagram are drawn simply for clarity, and are not meant to suggest a division into distinct groups. The points as plotted at this point represent a continuum in a uniform space. It is true that van der Rohe,

Le Corbusier and Wright can be considered to have quite different styles, as can different buildings by Wright alone, but proximity in this feature space is meant to suggest those buildings that are more similar to one another by degrees. The only significant outliers in this regard seem to be those caused by typology: the private houses vs. the public buildings, but at this point no attempt has been made to draw the distinction. (The machine learning algorithms to be described in Section 4 will allow this.) The fact that buildings of similar styles do fall near to one another in the reduced feature space confirms that the features indicated by the principal component are at least statistically related to style. Archetypes based on such a space may be used as style descriptors.

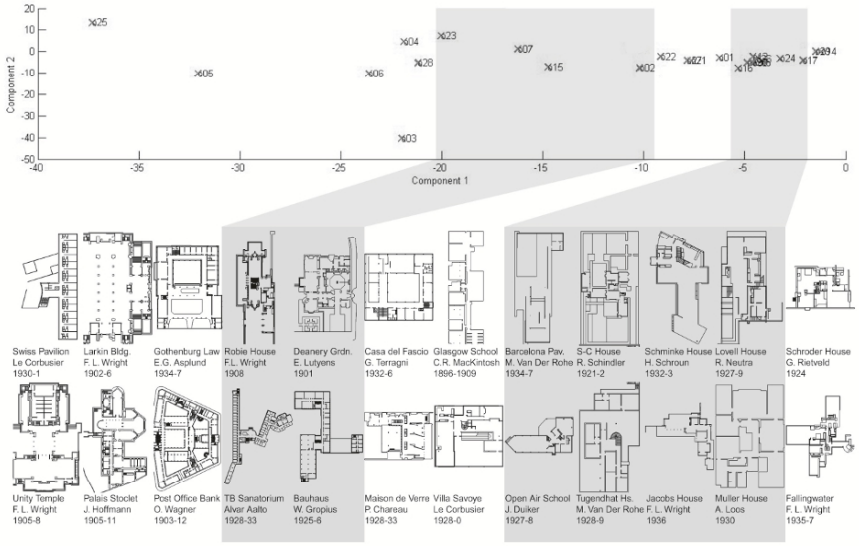


Figure 2. The example buildings plotted in feature space.

3.2. CLASSIFICATION OF PLANS INTO DISTINCT SETS

The plans above form a sort of continuum as they are taken arbitrarily from various styles over a forty year period, but the same process of graph analysis and dimensionality reduction can be used to plot the features of a particular group, and thereby define it. The above method was used again on a more focussed set of only two contrasting types in which there could be no ambiguity as to the label of each.

A set of 40 sample plans, Figure 3, was used containing examples of two building types: modern offices, and neoclassical museums. In overall shape the instances of these two types are quite similar to one another.

3.2.1. Results of plan classification by axial line analysis

In the plot, Figure 4(a), PCA is performed on the spectra derived from the axial graphs, and the results plotted against the first two components (again, derived statistically from the data). It can be seen that the two groups of plans can be almost entirely classified even on only the first principal component (horizontal axis). There is a clear separation into two clusters, with a large margin between. The museums, marked by '○'s, are distributed over a region to the left of the origin, and the offices, marked by blue '×'s, are toward the right. An outlier from the office group can be accurately classified as well using another dimension of the second principal component. A multi dimensional scaling method, Figure 4(b), was also used as an alternative to PCA, resulting in an even clearer distinction.

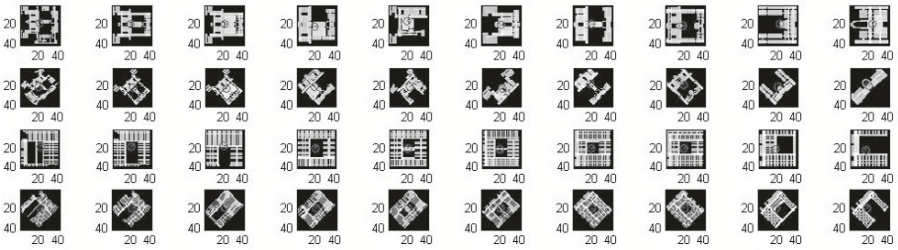
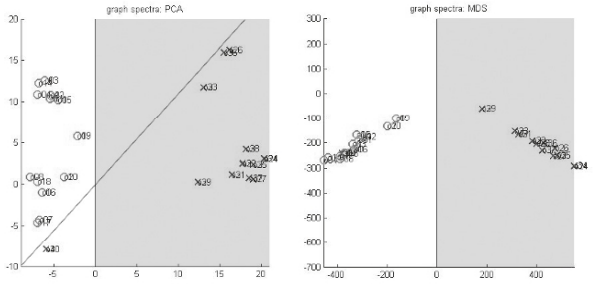


Figure 3. 20 plans representing museums (*upper*), and 20 offices (*lower*).



(a) (b)

Figure 4. The plans are classified by (a) the machine by PCA and (b) by multi-dimensional scaling (*right*). The horizontal axis is the principal axis of the data set; the vertical axis is the second.

3.3. DEFINING ARCHETYPES BY THE CLUSTERS

The resulting plots show points in space that are each representative of the spatial and perceptual features of a building plan. It is therefore possible to quantify such features as spatial relationships in a meaningful way. There is

variation in their placement, but they fall into two clear groups corresponding to the two classes of buildings used.

The setting of an archetype from this requires only the selection of the point that best describes the region of reduced feature space in which the examples of that style lie. Nearest-neighbour algorithms for analysis define prototypes as the generating points of the Voronoi tessellation that separates the classes (Duda et al. 2001), but for the synthesis to follow in Section 4 this would bias the result when these lie close to the edge of a cell. Also, for only two classes there are infinitely many such points. The point that minimises the distance to all examples in the cluster is simply the *mean*, which can be applied as easily for two style clusters or two thousand. This mean and the mapping to the reduced feature space together constitute the archetype.

The lower dimensional feature space that results allows a description of style that is convenient for analysis and measurement – in that any plan example can be evaluated, and compact – in that only a few dimensions need be used. Because most of the dimensions have been removed the space itself comprises only those features that are relevant to differentiate examples of one style from another, and the mean point of each of the clusters above can be said to be the archetypal ideal of each group.

4. Synthesis: Production of New Designs

The feature space and point that together define an archetype can be clearly used to measure example designs. In this section more sophisticated classification algorithms are used in place of PCA to derive the features, and methods for improving stylistic fidelity will be investigated. The use of *supervised* learning with *labeled* examples will imply a reduced feature space that is not just statistical, but meaningful. Analysis will also be combined with a generative method to synthesize new designs, and these will be used to evaluate the success of the archetype representation.

4.1. DESIGN AS SELECTION OF AFFORDANCES

Style can be considered a choice between given alternatives (Gombrich 1960), but rather than seeing this as a choice between generative rule systems, it can also be a choice of design moves within a system. While it may not be conscious (Goodman 1975) the act of creation implies the repeated choice of one design decision over another. One might consider this the ongoing selection of what Gibson (1979) terms *affordances*: opportunities for action provided by the environment, or in this case the design, as the design evolves over time. At any stage of the design of a building there are only certain possibilities open to the architect, and the act of adopting one style over another can be seen in this sense as a selection from the afforded alternatives. Tang and Gero (2001) suggest the act of

sketching, with constant drawing and re-evaluation is such a process, choice of rules is the inherent design activity of shape grammars, and the explicit representation of choices as design spaces has also been proposed for CAD environments (Brockman and Director 1991). In this section a generic system will be used to generate designs, but the style will be expressed by the choices made within it.

4.2. A BASIC OPEN GENERATIVE SYSTEM: BUILDING AGGREGATION

No generative algorithm is a direct counterpart to the axial graphs used in Section 3, but there is one that precedes them. Axial analysis was developed initially to analyse the variation in settlement layouts, particularly in French and English stylistic variants of what was termed the 'beady ring' form (Hillier and Hanson 1984). This form itself was found to be the natural result of a very simple but open generative system of aggregation, the basis of which will be used to produce new design examples in this section.

A minimal building unit is made up of two face-wise adjacent squares of a grid, with a closed building cell facing on to an open void space in front. The model allows these pairs to aggregate such that each new pair must join its open cell to at least one other open cell already placed, and the closed cell does not join another closed cell only at the vertex. The choice of available positions and orientations of each new unit is completely random, but each time the model is run, the emergent global structure forms that of the beady ring settlements studied, with a chain of open spaces onto which inner and outer groups of buildings face.

More important for the question of style as differentiation are the specific differences between towns. In Hillier and Hanson's own study they note the differences between the beady rings of France, and their counterpart villages in England that tend toward a more linear arrangement. These cultural differences in global form are also a result of the same uncoordinated local actions over time, yet the decisions of building placement that lead to a circular or a linear arrangement seem somehow to have been instilled into the individual members of the culture, not as contrasting sets of rules but as contrasting choices of application of the same rule set. "It is not simply the existence of certain generators that gives the global configurational properties of each individual [design]. It is the way in which variations in the application of the generators govern the growth of an expanding aggregation." (Hillier and Hanson 1984).

Although initially applied to town formation, this aggregation model is sufficiently general to represent rooms and corridors in a building or desks and chairs in an open plan office, depending on scale. While it uses a simple and constrained morphology, the grid is still able to represent configurations of very different patterns by the choices made in aggregation, and so it can

stand as an analogy to more sophisticated generative methods to demonstrate the principles of this paper. For this reason and for its historical roots in the development of the axial analyses in section 3, the overall structure of this grid aggregation model will be used below to show that stylistic norms can be learned from examples and used to emulate the original pattern.

4.3. TWO STYLES OF AGGREGATION AS EXAMPLES

Two artificial stylistic norms were chosen to be easily distinguishable from one another, and a simple algorithm written to aggregate open/closed pairs of units in the manner of each. The first is a strict arrangement of straight rows rather like highly planned settlements such as Manhattan, and the second is a random arrangement of units joined open cell to open cell, Figure 5.

To learn the two ideals, a classification algorithm is trained on the units as they are built. While the perception of spatial qualities of existing building plan examples in section 3 required the construction of axial graph matrices, this simplified grid model allows samples to be taken directly. Each time a new pair is placed in the plan, its relationship to the 7×7 cell neighbourhood surrounding the open half of the doublet is taken as its input. The 49 cells, each containing either a closed building (indicated by a filled square or 1), a public open space (a dot or -1) or yet unbuilt (empty or 0) are used as the computer's sensory experience of that particular built example.

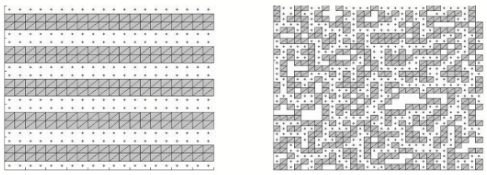


Figure 5. Two styles: Strict rows and random aggregation.

As in the case of the plan graphs, these neighbourhoods are points in an initial feature space. Each unique example can be represented by a point in a 49-dimensional space, a 2-dimensional projection (by principal components) of which is shown in Figure 6. Neighbourhoods of the straight rows are indicated by '×', and the random style by '○' markers in the centre. Clear clusters are less visible than in Section 3, but this can be overcome through the used of supervised learning algorithms in the following sections to perform the mapping to lower-dimensional feature space.

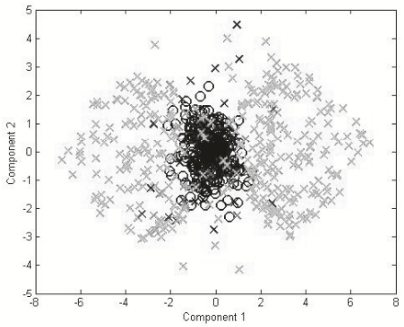


Figure 6. Examples from a 49-dimensional feature space.

In this space, the mean point of the cluster will represent an archetype of that style to be used in a straightforward building algorithm. At every step a given number of positions and orientations are available to be built, and the decision is simply the act of choosing which one of these affordances best fits the ideal by measuring proximity to the lower-dimensional archetype.

4.4. LEARNING AND BUILDING TO AN ARCHETYPE

Three experiments test the method of learning an archetype from examples and building to that archetype. The first tests that the style can be learned, with the hypothesis that clearer clustering will lead to a better resulting generation of the style. The second reveals that results can be improved by using a unique feature space reduction for each archetype. The third tests the hypothesis that the results of construction are independent of the choice of learning algorithm and particular representation of the archetype.

4.4.1. Clustering in a feature space and clarity of style: Training by Support Vector Machine

A crucial hypothesis to be tested was that the results of learning would allow designs to be produced in a given style. It implies there should be a direct correlation between clear clustering in the feature space and the strength of the style in the resulting design. A support vector machine (SVM) (Vapnik 1995) was used for the initial classification because its easily tuneable parameters allow its mapping tolerance to be adjusted to test this hypothesis.

SVMs operate by finding a maximally separating hyperplane between the two labelled classes in a higher dimensional representation of the input, and that representation is given by a non-linear function with a parameter that can be used to adjust the fit to the data – in this case the width of a Gaussian. Figure 7 shows the results for σ^2 = 5, 15 and 25 respectively. The SVM output is plotted in the left column with row examples to the left of random examples, such that the vertical axis represents the single dimension of SVM

output. The effectiveness of the classification is indicated in the second column images by the shading of each sample, where samples near the mean of the rows style are shaded in light grey and those of the random style are black. Clearly there is a clearer classification for the higher values of σ^2.

It is evident from the results that as σ^2 increases, the cleaner separation between the two groups by the algorithm results in a clearer construction, as shown in the images to the right. At each construction step, the possible sites and orientations are evaluated by the SVM, and the one closest the mean of either style as learned is selected. The completed constructions over a period of time are shown, one emulating the rows style as learned and the other the random arrangement, and the overall patterns are most easily seen for σ^2 = 25, particularly for the straight rows.

The initial hypothesis is confirmed, but the separations in Figure 7 are never quite enough, and the classifier can only produce adequate rows with an artificially created set of 'perfect' examples of row neighbourhoods. These are all identical, so that each is exactly perceived as the ideal archetype, and consequently the perfect classification of the two groups results in a stronger expression of the style, Figure 8.

4.4.2. Clarifying the archetype feature space: Training by neural network

The method thus far performed one analysis for the principal components of all styles. It would yield appropriate archetype definitions if all styles differed in the same features, and thus could be classified in the same space, but this is unlikely. Rather than merely classifying two styles, the benefit of the clear archetype in Figure 8 suggests the choice of a feature space fit to a single style yields stronger results. In this section a unique feature space is found for a single style by training a neural network to find a space in which the points are clustered closely together as differentiated from all others.

A neural network was used to learn the rows style only, with the random examples serving as mere background from which to differentiate the relevant features. A Feedforward Multilayer Perceptron (MLP) (Rosenblatt 1958) was used, with 49 input nodes corresponding to the state of the neighbourhood, 50 nodes in the hidden layer, and a single, linear output that rates each example. Training was conducted by exposing the network to 450 examples from each of the two test styles and backpropagation of errors. Because the goal is to learn the features of the rows style only rather than to classify both, a variation on the typical error function was used.

As there was no need for a target for examples outside the style in question however, the target for the rows was set to 0, and the reciprocal of the error used for all other examples, causing the error to fall as examples appear farther away. The advantage of this modified error function was a large separation and an output for most samples very close to 0.

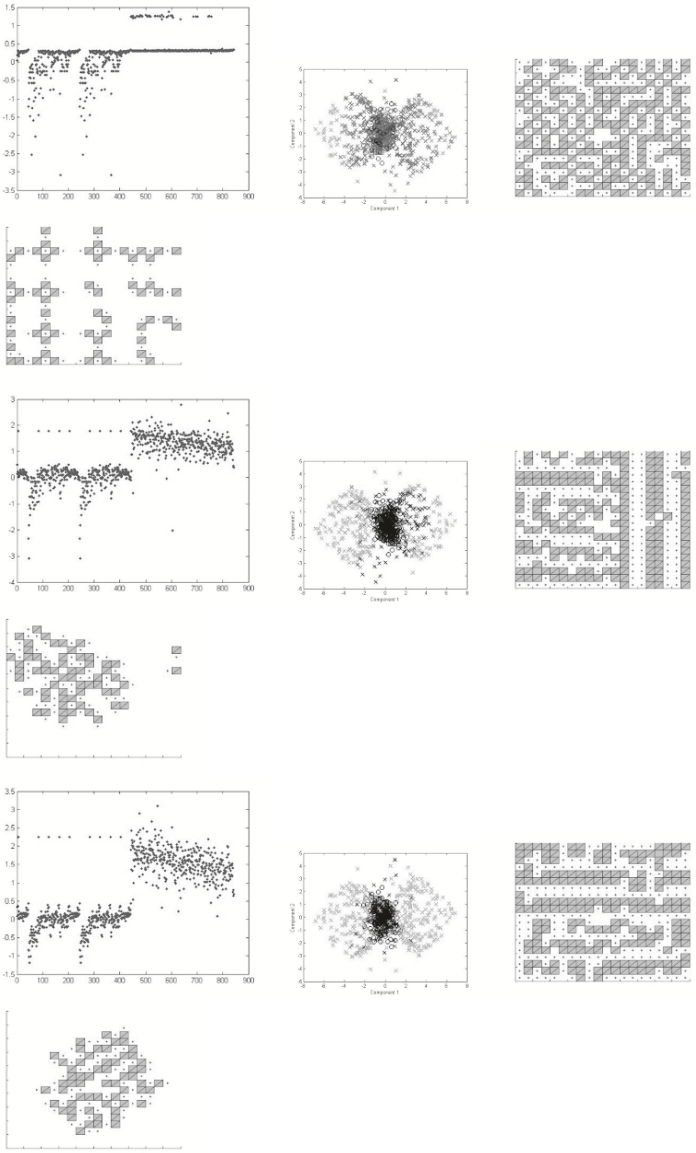


Figure 7. Building results for algorithms trained with a SVM: $\bar{\sigma}2$ = 5 (top), 15 (centre) and 25 (bottom). The first image on the left shows the mapping of 900 examples against the vertical axis. The second indicates apparent membership in each cluster by the shading of the points. Resulting building patterns follow emulating rows, then random aggregation.

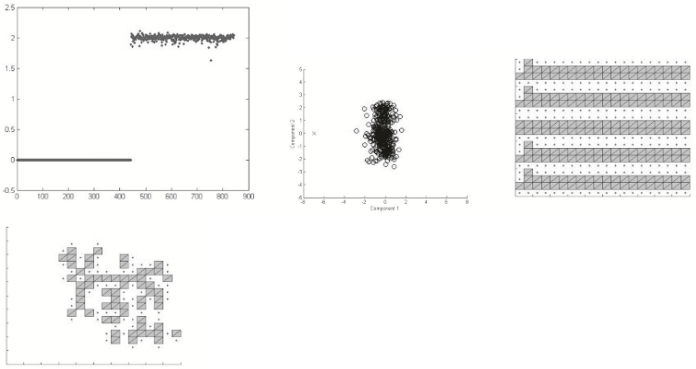


Figure 8. The same training on a set of 'ideal' examples.

Results of this style-specific feature space were superior to those of the SVM in Section 4.4.1. In Figure 9 each of the examples is shown as a single dot in the vertical axis corresponding to the value of the network's single node output. After training, most of the first 450 examples along the horizontal axis (the row units) appear at 0, and most of the others (the random aggregations, to the right) as far away (note the extreme scale of the output axis). The resulting aggregation of open and closed cells produced by the building algorithm very closely resembles that of the original rows from which it was trained.

4.4.3. Variations on the representation to learn the same style

Because the style is described by a feature space rather than symbolically, the actual method of feature space mapping in the archetype is quite arbitrary. This section tests that it can be changed and still lead to recognisable output. Interestingly, like Gombrich's game of 'Twenty Questions' in which the initial questions can also be arbitrary, the choice of classification algorithm used to define the style does not appear to matter. In fact one style can be described in many different ways, or feature spaces of different dimensionality.

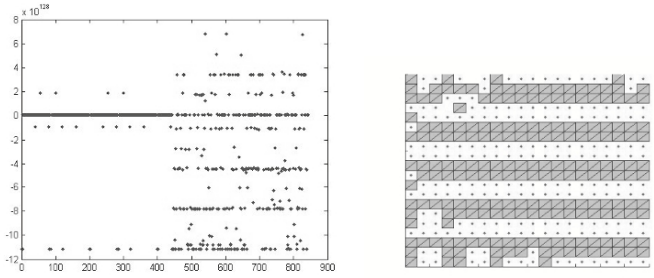


Figure 9. Training of a three layer network on the row samples.

Figure 10 shows the result of several very different learning algorithms exposed to the same set of examples, each resulting in a very different mapping of features (left) but very similar overall construction of rows. First is a neural network similar to the one in Figure 9, except that only the nearest examples of the random style were used in training. Below this, a different technique is used to train the network: errors from both groups are measured from the mean, but rather than adding the weight updates at each step for the examples from the random style, they are subtracted. The last example is a differently structured network entirely: a Kohonen self-organising feature map (Kohonen 1982). The subtraction training and the Kohonen feature map were found to be the most successful at replicating the overall pattern for this test style.

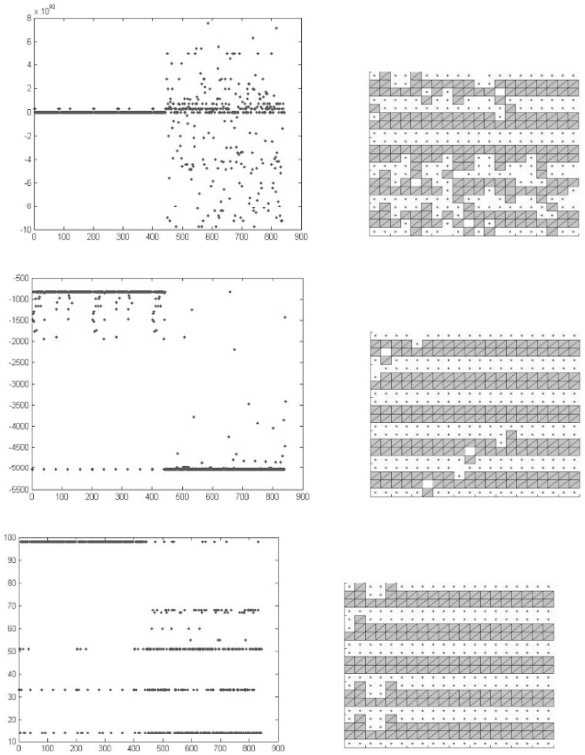


Figure 10. Three completely different algorithms (two double-layer neural networks and one Kohonen network) result in different feature spaces (left), but make similar evaluations and similar constructions.

The similarity of the final constructions indicates a style can be represented many different ways. Even with the constrained grid morphology of design space, there is a drastic difference in the feature

spaces, Figure 10, left. These networks have different structures and different training methods, but produce similar results to one another. Feature spaces may differ in detail and even dimensionality, as each of the algorithms is capable to mapping to an arbitrary number of dimensions. The only necessary common process is that each forms an archetype in its unique feature space based on the examples of that group.

5. Conclusion

The idea of a style in any discipline is a fluid concept that is always subject to change, and therefore suited to a flexible representation. What is suggested here is that it can nevertheless be accurately represented and emulated. This work has presented an algorithmic method for both deriving a stylistic definition automatically from examples, and using it to generate new designs. Architectural examples were used, and were investigated primarily in terms of their spatial features, but it is intended as a general model in that other forms of input and classification algorithms may be used. Likewise, axial analysis and the aggregation model are not essential to the method, but the principles of feature space reduction and archetype should apply to a variety of analysis and synthesis techniques.

The concept of the archetype proposed is of a defined ideal and of a space in which to measure example designs. It contains only the features most relevant to define that style, but they are not counted as symbolic wholes. Instead one can measure an example's similarity in degrees, on an objective and continuous scale. This results in a definition of style that is flexible, can evolve, and is based on examples.

While fixed, rule-based systems are used as design aids by generating new examples of designs, a flexible, example based method such as this would assist in a very different way. While the archetype may be resistant to symbolic description, so very often are our own mental processes of style recognition, and in many complex problems we can more easily communicate by example than by explicit description. By automatically generalising its representation based on examples presented to it by a designer, such a design aid may propose output based not on rational clarity of process, but on the simple choice of precedents, fashion, taste or a hunch.

The definition of style provided by the archetype is analytical rather than generative, but there is still an obvious role for generative systems to play. The aggregation model in Section 4 was chosen for its simplicity and common origin with the analysis in the previous section, but shape grammars and other generative rules could be applied – a likely avenue for future exploration. Their role in this regard however, is as a framework for exploration of many styles rather than a definition of one.

Creative design is ultimately not a matter of rule following, but of judgement, and the model presented here proposes the flexibility this implies may extend to the definition of styles themselves.

Acknowledgements

I would like to thank professors Alan Penn and Philip Steadman for space syntax related advice, and for introducing some of the background examples presented in this paper. This research has been supported in part by the Engineering and Physical Sciences Research Council, UK and by Foster and Partners.

References

Alexander, C, Ishikawa, S, Silverstein, M, Jacobsen, M, Fiksdahl-King, I and Angel, S: 1977, *A Pattern Language*, Oxford University Press, New York.

Brockman JB and Director SW: 1991, The Hercules task management system, *Procedings of the International Conference on Computer-Aided Design*, pp. 254-257.

Chan, CS: 1994, Operational definitions of style, *Environment and Planning B: Planning and Design* **21**: 223-246.

Dalton RC, and Kirsan C: 2005, Small graph matching and building genotypes, *Environment and Planning B: Planning and Design* (to appear).

Desyllas, J: 2000, *The Relationship between Urban Street Configuration and Office Rent Patterns in Berlin*, PhD thesis, Bartlett School of Graduate Studies, UCL, London.

Desyllas, J and Duxbury E: 2001, Axial maps and visibility graph analysis, *Proceedings, 3rd International Space Syntax Symposium,* Georgia Institute of Technology Atlanta, pp. 27.1-27.13.

Duda, RO, Hart, PE and Stork DG: 2001, *Pattern Classification*, John Wiley, NY.

Durrant-Whyte H: 2004, Autonomous navigation in unstructured environments, *Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision*, pp. 1-5.

Gero, JS and Kazakov, V: 2001, Entropic-based Similarity and Complexity Measures of 2D architectural drawings, *in* JS Gero, B Tversky and T Purcell (eds), *Visual and Spatial Reasoning in Design II*, Key Centre of Design Computing and Cognition, Sydney, pp. 147-161.

Gibson JJ: 1979, *The Ecological Approach to Visual Perception*, Houghton Mifflin, Boston.

Gombrich E H: 1960, *Art and Illusion*, Phaidon, London.

Goodman N: 1975, The status of style, *Critical Inquiry*, Volume 1, Reprinted in Goodman N: 1978, *Ways of Worldmaking*. Hackett Publishing Company Inc. Indianapolis.

Hersey, GL and Freedman, R: 1992, *Possible Palladian Villas: (Plus a Few Instructively Impossible Ones).* The MIT Press, Cambridge MA.

Hillier B, Hanson J, Peponis J, Hudson J and Burdett R: 1983, Space syntax, *Architects Journal* **178**(48): 67-75.

Hillier B and Hanson J: 1984, *The Social Logic of Space*, Cambridge University Press.

Hillier B, Penn A, Hanson J, Grajewski T and Xu J: 1993, Natural movement, *Environment and Planning B: Planning and Design,* **20**: 29-66.

Hillier B and Shu S: 2001, Crime and urban layout: The need for evidence, *in* S Ballintyne, K Pease and V McLaren (eds), *Secure Foundations: Key Issues in Crime Prevention and Community Safety*, IPPR, London.

Jupp J, and Gero, JS: 2003, Towards computational analysis of style in architectural design, *in* S Argamon (ed), *IJCAI03 Workshop on Computational Approaches to Style Analysis and Synthesis*, IJCAI, Acapulco, pp. 1-10.

Jupp, J and Gero, JS: 2006, A characterisation of 2D architectural style, *Journal of the American Society of Information Science* (to appear).

Knight TW: 1998, Shape grammars, *Environment and Planning B: Planning and Design,* Anniversary Issue, pp. 86-91.

Kohonen T: 1982, Self-organized formation of topologically correct feature maps, *Biological Cybernetics* **43**(1): 59-69.

Koile, K: 1997, Design conversations with your computer: Evaluating experiential qualities of physical form, *CAAD futures 1997*, pp. 203-218.

Koile, K: 2004, An intelligent assistant for conceptual design, *in* JS Gero,(ed)*, Design Computing and Cognition '04*, Kluwer, Dordrecht, pp. 3-22.

Koning H and Eizenberg J: 1981, The language of the prairie: Frank Lloyd Wright's prairie houses, *Environment and Planning B: Planning and Design* **8**: 295-323.

Peponis J, Hadjinikolaov E, Livieratos C and Fatouros DA: 1989, The spatial core of urban culture, *Ekistics* **56**(334/335): 43-55.

Prats M, Earl C, Garner S and Jowers I: 2006, Exploring style through generative shape description, *AIEDAM Journal* **20**(3): (to appear).

Rosenblatt F: 1958, The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review* **65**(6): 386-408.

Spiliopoulou G and Penn A: 1999, Organisations as multi-layered networks, *Proceedings, 2nd Intl. Space Syntax Symposium*, pp. 1-24.

Stiny G: 1976, Two exercises in formal composition, *Environment and Planning B: Planning and Design* **3**: 187-210.

Stiny G: 1980, Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design* **7**: 343-351.

Stiny G and Mitchell WJ: 1978, The Palladian grammar, *Environment and Planning B: Planning and Design* **5**: 5-18.

Tang H-H and Gero J: 2001, Sketches as affordances of meanings in the design process, *in* JS Gero, B Tversky and T Purcell (eds), *Visual and Spatial Reasoning in Design II,* University of Sydney, Australia, pp. 271-282.

Tillmann, B, Abdi, H and Dowling, WJ: 2004, Musical style perception by a linear auto-associator model and human listeners, *Proceedings of the 8th International Conference on Mucic Perception & Cognition*, Evanston, IL.

Turk, M and Pentland, A: 1991, Eigenfaces for recognition, *Journal of Cognitive Neuroscience* **3**(1): 71-86.

Turner, A: 2005, An algorithmic definition of the axial map, *Environment and Planning B: Planning and Design* **32**(3): 425-444.

Turner A, Doxa M, O'Sullivan D, and Penn A: 2001, From isovists to visibility graphs: A methodology for the analysis of architectural space, *Environment and Planning B: Planning and Design* **28**(1): 103-121.

Vapnik V: 1995, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.

Weston, R: 2004, *Key Buildings of the Twentieth Century: Plans, Sections, Elevations*, WW Norton & Co., New York.

A DIGITAL BRIDGE FOR PERFORMANCE-BASED DESIGN

Simulation of building physics in the digital world

DIRK A SCHWEDE
The University of Sydney, Australia

Abstract. The augmentation of digital design representation with programmed analysis capabilities can result in a shift from structure-based to performance-based designing. This paper presents a system to translate a simple digital structure representation into information about the multi-dimensional highly integrated and dynamic physical behavior of the design object. The user interface uses an object-oriented representation familiar to the designer, while the physical behavior is calculated internally with an abstract and space-based model formulated in form of a constructive language. The system is intended to serve as a "digital bridge" in the circle of design activities to enable performance-based designing.

1. Introduction

1.1. DIGITAL AUGMENTATION

Digital augmentation of design representation with programmed analysis capabilities can support the architectural process by translating the structure of the design object into more complex property and behavior information, than can be obtained by human reasoning alone.

Such information can support a more integrated understanding of the behavior of the of the design object, so that knowledge about the designed structure and about its actual behavior (performance) would become closer connected in the analysis and evaluation process. This could result in a shift from a structure-based to a more performance-based design approach.

Figure 1 shows the design process in the notation of Gero's FBS-framework (Gero 1990). In extension of Gero's set of design activities, the simulation model generation, as it occurs in the augmented design process, is included in right depiction. The performance is represented by the set of actual behavior information Bs in the original notation of the framework.

J.S. Gero (ed.), Design Computing and Cognition '06, 23–40.
© 2006 *Springer. Printed in the Netherlands.*

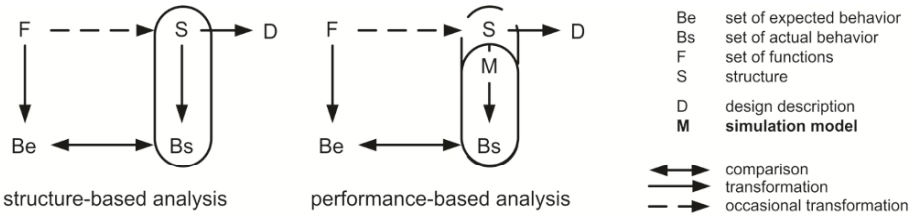


Figure 1. Left: analysis the structure-based, not augmented (Gero 1990) and right: in the performance-based, augmented design process.

The figure shows the analysis activity in the structure-based design process in comparison to the analysis in the performance-based design process. The prediction of the structure's actual behavior is not based on the analysis of the structure alone but on information about its performance, conceived by computational simulation of the structure's properties and its dynamic behavior interacting with its environment.

In the digital design process the simulation model generation is as far as possible a programmed and automated activity on basis of a digital structure representation, in a form understandable and editable by the designer. While the simulation is far beyond, what the designer can achieve by reasoning, the display of the results is near the designer's experience and domain language. Thereby simulation functions as a "digital bridge" in the circle of the performance-based design process.

This paper is concerned with the digital support of the analysis activity in the human design process. It is not concerned with automatic evaluation and the digital generation of design suggestions (synthesis).

This paper describes a digital system, which translates a three-dimensional structure description into information about its dynamic physical behavior using a highly integrated and self-contained physical model to represent physical phenomena required for comfort quality assessment of buildings.

1.2. COMFORT ASSESSMENT REQUIREMENTS

A literature survey on available comfort models (Schwede forthcoming), using the building-in-use post-occupancy-method (Vischer 1989) to define the scope, revealed that the assessment of physical aspects of comfort requires a three-dimensional, highly integrated, simultaneous and dynamic representation of the following phenomena:

- temperature,
- moisture content,
- CO_2 content,
- odorous substances content,
- velocity,

- light, and
- sound.

1.3. AVAILABLE SIMULATION TOOLS

Such highly integrated models of phenomena and their dynamic behavior are not available at the current stage. Building simulation implementations represent physical phenomena required to assess the thermal conditions in node- and building-zone-based models with a low resolution of the physical phenomena.

Models of other physical phenomena, such as light and sound, might be available in simulation tool suites, which provide assess to common modeling and result display functions, but which operate several separate or loosely coupled simulation engines on basis of one building model database (e.g. Marsh 1997; ESRU 2002). More integrated simulation models are constrained to specific views and two-dimensional representations (e.g. Künzel 1994) or to steady state calculations.

Integration of various domain models on basis of a common database is discussed as one of the key problems of the application of simulation in the design process. Mahdavi (1999) uses a space based-representation in the implementation of the simulation suite SEMPER in order to translate seamlessly between a shared object model and domain object models (thermal and flow model). In SEMPER the design object is modeled in a spatial grid (e.g. $1x1x1m^3$) and the grid cells are simulated similar to building zones in other simulation programs. Walls, for example, are represented as a linear sequence of nodes in the thermal simulation model.

In more recent work Suter and Mahdavi (2003) use a sheet representation (SHR) and a solid representation (SOR), in order to supply node-based models as well as space-based models (e.g. FEM-models) with information from the shared object model. They apply a space-based technique for the mapping between these representations.

Nevertheless the literature review on comfort assessment models, cited earlier (Schwede forthcoming), revealed that the integration and the simultaneous simulation of physical phenomena is required for the assessment of the comfort conditions, rather than the operation of separate simulation models on basis of a central data model. Therefore the research presented in this paper aims to integrate on the level of the simulation engine, rather than on the level of the data model.

1.4. PHYSICAL BEHAVIOUR

Physical behavior is a function of material in space and the physical state of the world. This behavior can be described with well-known physical laws, is modulated by physical properties of the space and driven by differences of

the physical state in neighboring areas. The physical state is the result of transport processes and storage phenomena (or of chemical reactions) at the location. Physical behavior is not a function of object-oriented concepts, which humans use to describe objects in the physical world.

Conrad Zuse (1970) discusses the idea of a calculating space and suggests using automata concepts to describe physical phenomena. He understands the calculating space as a theory to describe our world in the quantum paradigm alternatively to conventional physics. Fredkin (1992) argues even further for the paradigm of a finite nature, he assumes that at the end everything, including space and time, will turn out to be atomic or discrete and that there is a fundamental process of physics, which is computation universal. He states that there are no solutions in closed form, similar to the ones in physics and that nothing can be calculated faster with the same resolution than to do the simulation step-by-step.

However the reasonable size of a quantum to describe physical processes in architecture is compared with the scale Zuse and Fredkin suggest to explain the world as such, of macroscopic scale. The smallest length designed and plotted in architecture drawings is 1 mm. The fabrication accuracy on the building site is 1 cm for massive constructions and 1 mm for the steel work and fit out.

Nevertheless the understanding of physics as a system of interacting simple and discrete elements representing material in space, is (under application of digital technology) able to overcome complexity introduced in the simulation models by object-oriented representation concepts.

2. Concept

2.1. CONSTRUCTIVE LANGUAGE

The simulation model is developed in form of a constructive language to ensure its universal and context-independent applicability, so that various questions about the physical behavior of the design object (of which not all are known, when is model is developed) can be answered on its basis, to allow its processing on basis of various design representation and to enable a demand-oriented result display.

A set of basic spatial elements is created, which displays different meanings according to the physical state calculated for their locations, when the objects synthesized from them are looked at from different domain viewpoints. Not only the topology of the basic elements and the context determine the meaning of their combination, but also the multiple inherent properties of the elements and their conjunctions.

The constructive language to simulate physical behavior of material in space is defined by the following components and their concepts are explained in the following paragraphs:

- basic elements — congeneric cells with physically self-contained behavior map
- rules of topology — geometrically self-contained design space
- rules of validity — demand-oriented validity range
- meaningful context — evaluation models

2.2. GEOMETRIC MODEL

The geometrically self-contained design space sets the spatial boundaries in which the design takes place at the beginning of the design session. Initially it contains material with meaningful properties (e.g. air) and the conditions inside the design space and at its boundaries are defined (e.g. weather). Design activity changes the content of the design space by adding geometric objects within its boundaries, but does not extend it spatially.

The concept of geometrical self-containedness ensures that only designable model content of the geometric model has to be specified at design time. Not-designable model content can be determined automatically as the geometry is represented in a space-based and closed form.

2.3. SIMULATION MODEL

Congeneric cells represent volume and surface properties and storage phenomena, while congeneric conjunctions represent transport processes in the model. Together they represent the physical behavior of the design space.

The physically self-contained behavior map represents the dynamic properties of the cells' material as a function of the cells' material and its physical state. The simulation model is valid for physical states within the validity range.

2.3.1. Physical Self-Containedness

The concept of physical self-containedness represents the fact, that the physical behavior of the world is a result of an inseparable system of interrelated and simultaneous physical processes. The model is based on a set of well-known physical laws and complete and integrated enough to calculate the system of physical phenomena without any implementation of their interaction at design time.

2.3.2. Congeneric Cells

The design space is dismembered into a three-dimensional structured grid of cubic cells. Physically self-contained behavior maps for each cell volume are allocated, according to the material at the cells' locations. The congeneric cells' behavior is well defined by physical laws and only dependent on the cells' properties, their history and their boundary conditions. The cell properties are dynamic functions of the cell's geometry, material and its physical state, Figure 2.

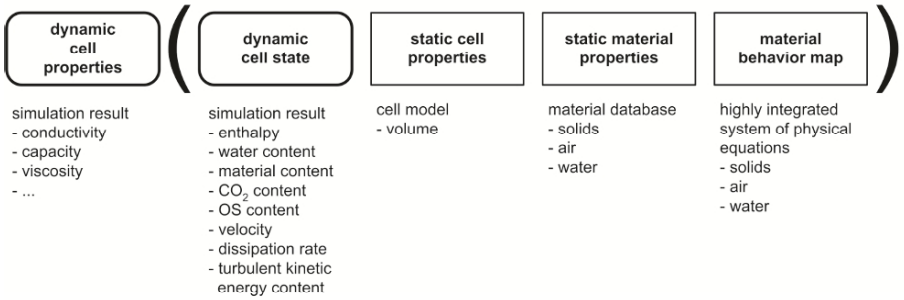


Figure 2. Calculation of dynamic cell properties.

The physical state is either a result of the previous condition or it is set to a designed value at the beginning of each time step, if an activity property is allocated to the cell. The concept of activities allows modeling sources and sinks of the represented physical phenomena, such as for example light emitter or heating elements.

At any point in time a cell only "knows" its own state and properties. The boundary conditions of each cell are given by the physical state of their neighboring cells. The interaction between two cells is modeled by congeneric conjunctions.

2.3.3. Congeneric Conjunctions

Congeneric conjunctions represent exchange between the cells or between the cell surfaces. Near-conjunctions connect spatially adjacent cells. Remote conjunctions connect the cell surfaces that can "see" each other but are separated by transmitting material (e.g. air) between them, Figure 3.

The mathematical formulations of the conjunctions are simple and of common structure, for the various processes, so that transports of various kinds can be calculated with a, as far as possible, common algorithm. The structure of the mathematical formulation is depicted in Figure 4.

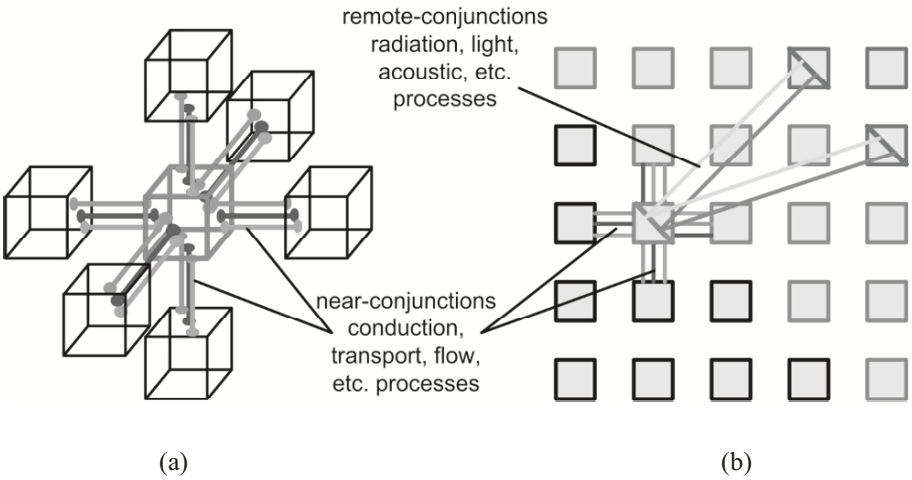


Figure 3. Concept of congeneric conjunctions:
(a) near-conjunctions, and (b) remote-conjunctions.

The near-conjunction primary processes represent conduction, diffusion and flow phenomena. Convective transport phenomena associated with these processes are modeled as secondary processes. Heat radiation, light and sound are modeled as remote-conjunction processes.

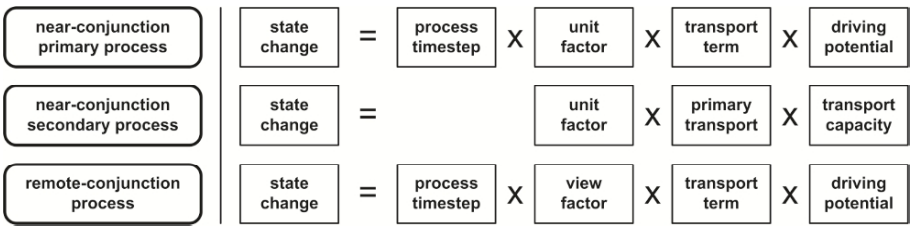


Figure 4. Structure of the mathematical formulation of conjunction processes.

As example the data structure of the transport processes of near-conjunctions between two cells are shown in Figure 5. A near-conjunction connects two cells and contains a vector of process datasets.

Each process dataset is connected to the state variable and the driving state of the process it represents in the cells' datasets. It contains the transport term (resistance) and a process-individual calculation timestep (frequency).

The transport term is calculated as a function of the dynamic properties of the two cells and the cells' geometries. The process-individual timestep is calculated each timestep as a function of the transport capacity of the cells and the transport resistance of the process dataset in order to avoid oscillation of the calculated cells' states.

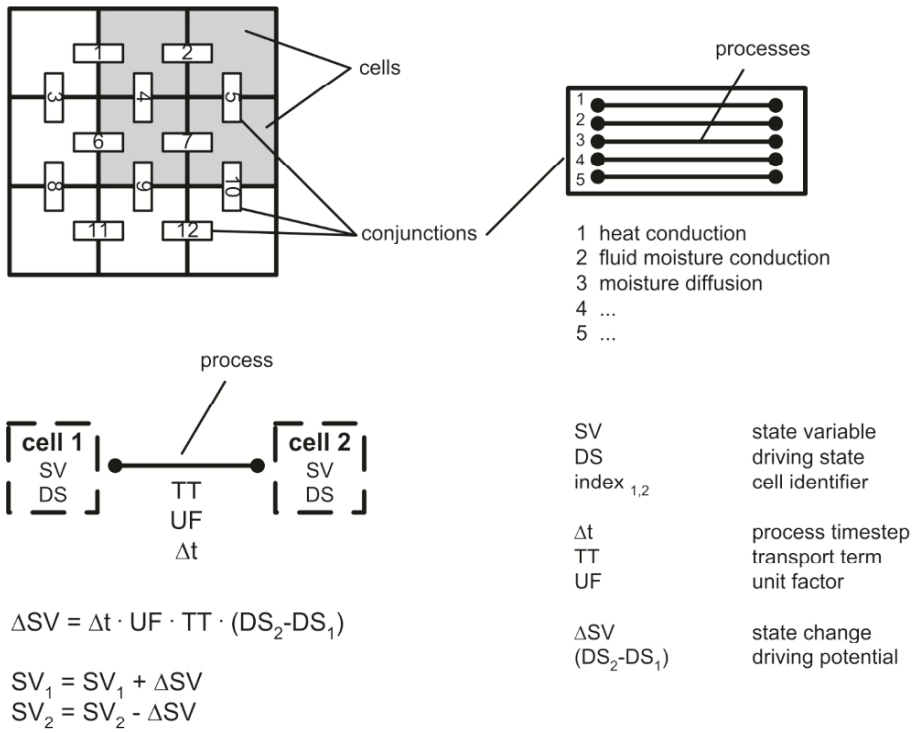


Figure 5. Depiction of the data structures of the model (cells, conjunctions, processes).

2.3.4. Simulation Engine

For the simulation of the structure's dynamic behavior the process datasets of each conjunction are copied into the central process list as depicted in Figure 6. The process-individual timestep Dt and the number of calculations C per overall-timestep DT are determined. The counter variable c is set zero at the beginning of the overall-timestep.

The algorithm steps through the process list repetitively (multiple times during one overall-timestep) during the simulation. The variables cc and C_{max} of the list and c and C of each process are used to trigger the calculation of the individual processes at the appropriate moment during the simulation timestep as shown as C++-code in Figure 6. The process list is processed repetitively until all exchange events are calculated as often as required.

An exchange event changes the state of both cells engaged in the process. The cell and conjunction properties are updated at the end of each simulation timestep.

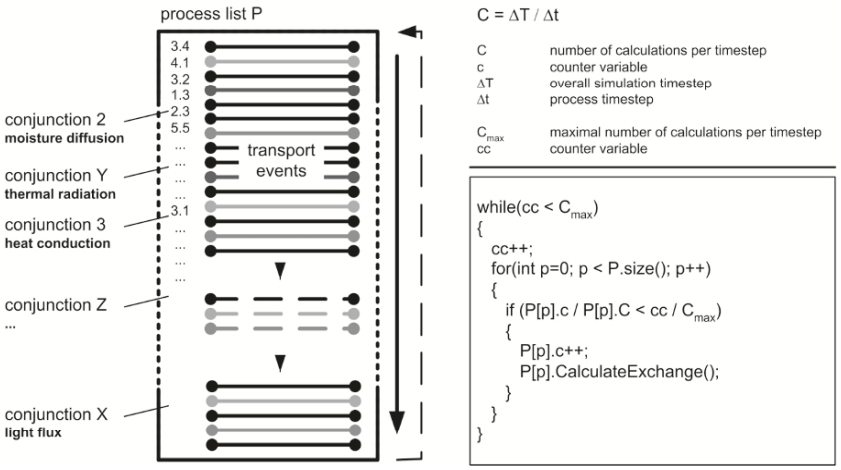


Figure 6. Repetitive calculation of the process list within one timestep.

2.3.5. Validity

The rules of validity specify the limits of physical parameters necessary to assess comfort and limit the calculation to the conditions the equation catalog of the cells' self-contained behavior map is valid for.

2.4. ACTIVE BEHAVIOR MODELING

Activity models are assigned to objects in order to make them a source or a sink of physical phenomena, such as a heat source or light emitter. They are modeled as constant values, as equations with simulation parameters and simulation results as input or their values are read from a file. During the simulation the state of an active cell or an active cell face is set to the designed value. The concept of activities allows modeling internal sources and sinks as well as the conditions at the boundaries of the design space, Figure 7.

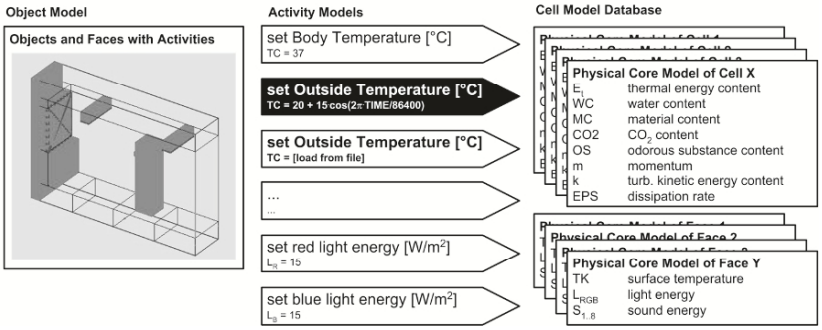


Figure 7. Modeling function for activities.

2.5. EVALUATION MODELS

The concept of evaluation models allows customizing the result display for the context of the investigation and individually for the demands of the investigators. Meaningful views on the design space's physical behavior can be synthesized by modeling evaluation models as equations with parameters of the simulation model and simulation results as input, Figure 8.

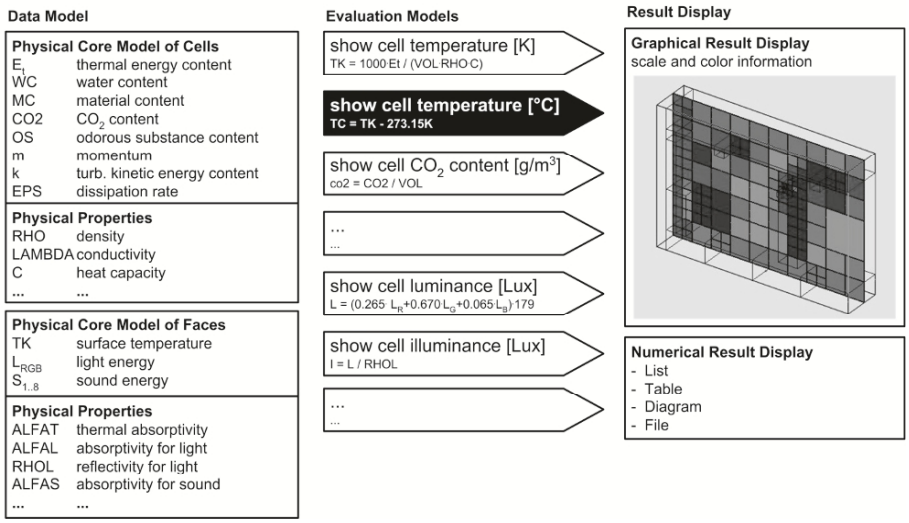


Figure 8. Formulation of evaluation models.

Sophisticated and integrated evaluation models can be formulated using several of the simultaneously calculated physical phenomena as input. A simple comfort evaluation model for the three-dimensional false color display could be formulated as, Figure 9:

		$T_{\circ C}$	<	20°C	too cold,	display in shades of blue
20°C	<	$T_{\circ C}$	<	26°C	comfortable,	display in shades of yellow
26°C	<	$T_{\circ C}$			too warm,	display in shades of red

While the three-dimensional false color display are applied for qualitative assessment, datapoints, surface sensors and balance volumes, which are modeled as virtual sensing objects in the geometric model, are used to read results for quantitative assessment from the simulation.

3. Implementation

The concept was implemented and tested on Pentium 4 notebook with 2.5GHz and 1GB memory in C++ in a Windows XP environment, using Borland C++ Builder 6 and OpenGL for the user interface implementation.

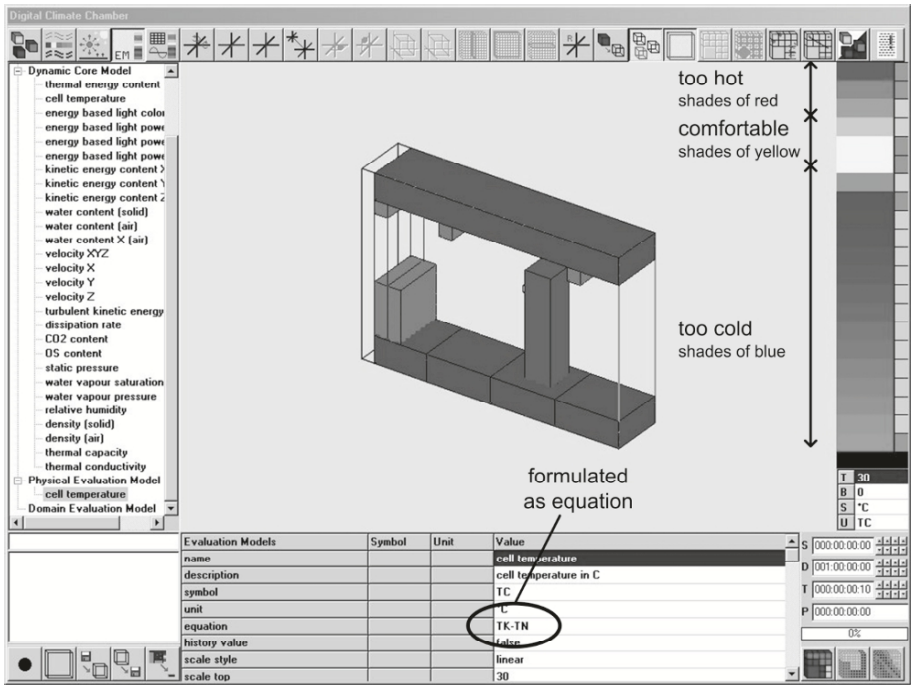


Figure 9. Screenshot of the user-interface for modeling evaluation models.

The prototype consists of three main parts the user interface, the model translator and the simulation engine, of which the user interface is used for the design functions as well as for the result output and evaluation. Figure 10 shows the schema of the prototype implementation.

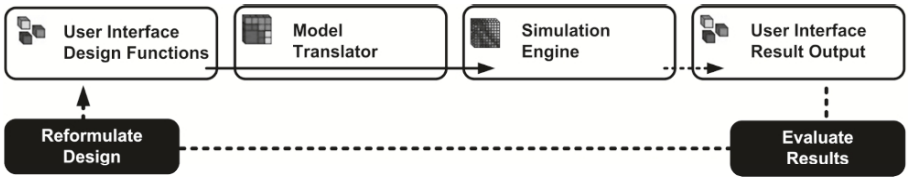


Figure 10. Schema of the prototype implementation.

3.1. DATA MODEL

Although a modeling interface to enter the geometric model was implemented in the system for the model development, the concept and the model translator algorithm would allow modeling with other design tools, which are able to provide the following basic information about the design object's geometry and properties:

• geometry	physical objects virtual objects (sensors)
• identifier of geometry-related properties (label)	material active behavior

Additional system specific information is required for the result output and the content of the material, activity and evaluation model database. This information is provided in project independent databases:

• display functions	evaluation models result output functions
• detailed property information	material activities

3.2. MODEL TRANSLATION

The model translation involves a sequence of steps from the object model entered by the user into the cell and conjunction model.

The object model is associated with a volume representation as shown in Figure 11. A volume is a tetrahedron, which's four points define four planes, each of which defines two halfspaces. Simple halfspace operations are applied in the successive steps of the translation algorithm (see C++ code in Figure 11) to test for example, if a test point is in or outside an object.

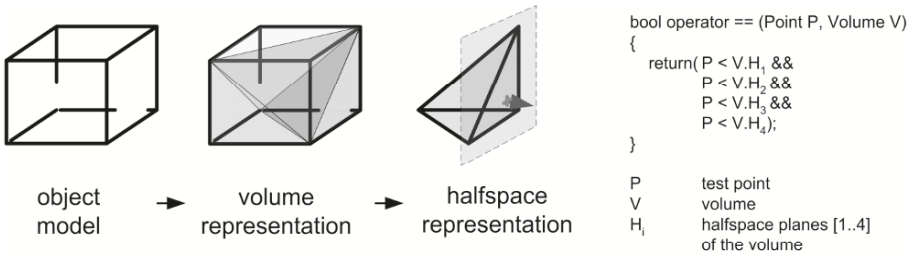


Figure 11. Representations of the object model in the prototype implementation and formulation of halfspace-operation as C++-code.

In Figure 12 the steps of the translation from the object representation into the structured-cell grid representation are depicted in step 1 to step 4. Step 5 depicts the near-conjunctions and step 6 the generation of the cell faces. The further steps (remote-conjunction generation, connection of sensors and activity datasets, setting of start values) are omitted as the detailed explanation of the translation process is beyond the scope of this paper.

Although the prototype does only allow designing with rectangular object, the concept of cells and cell faces would allow the representation of

objects with sloped faces. While the cells follow the rectangular shape of the cell grid and would partly overlap with the object's edges, the cell faces are oriented and positioned like the object faces they are associated with as sketched in Figure 3.

An algorithm for the translation of models with sloped faces was developed during the research. It was not implemented in the final prototype implementation due to long processing times and insufficient robustness.

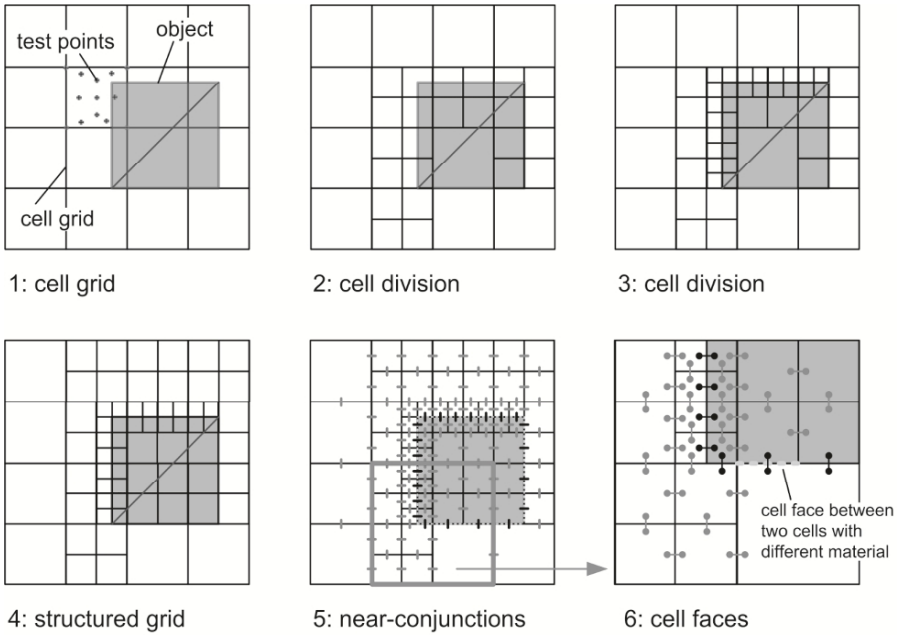


Figure 12. Translation of the object model into the cell model, generation of conjunctions and cell faces.

4. Application

Figure 13 shows the application of the self-contained model in the design process schematically. Figures 14 and 15 illustrate the process with a short sequence of screenshots, showing the "design" of an office room as application example.

Initially the design space is filled with air, Figure 14, (1.1), and then the outdoor conditions are added in form of an object with active behavior (1.2). Following the designer enters the first design suggestion (2.1) and translates the model in the cell representation (2.2, which would not be displayed to the designer in the design process).

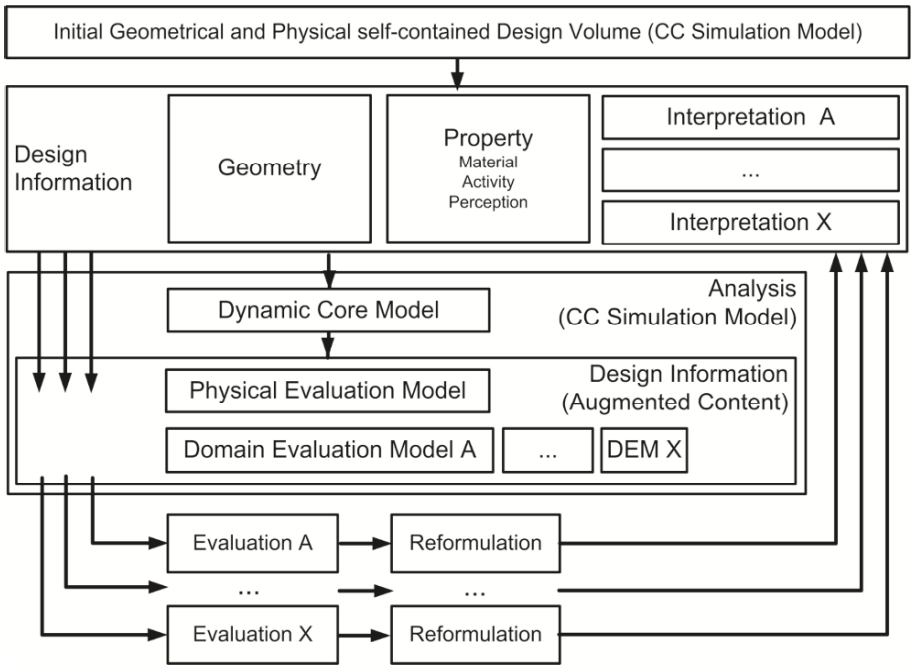


Figure 13. Application of the self-contained Model in the Design Process.

A first simulation would reveal that the temperature near the window is uncomfortable cold (2.3), and that the daylight conditions are acceptable (2.4), but that additional light is required in the depth of the room. More results of the integrated model could be reviewed to understand the shortcomings and qualities of the design proposal's performance.

At this stage a qualitative assessment is sufficient and the 3-dimensional depiction allows fast and detailed understanding of the problem.

In the following step the design is improved (3.1). A heating element is added under the window and light fittings are added at the ceiling. The light fittings are modeled as light sources, but do also introduce heat in the room.

Additionally an occupant is entered in the space (3.1) as heat and air contaminate source. Further sensing capabilities of the body to evaluated non-uniform thermal conditions and at the position of the eyes to assess glare are added.

Figure 15, 3.2 highlights the object with active behavior. After the changes of the geometric model are entered, the model is translated into the cell model. Plate 3.3 shows the remote-conjunction generation (which was also performed, but not displayed, for the first simulation run).

The second simulation shows, that the heating element and the heat emission of the other sources prevent the room from cooling down and that the light conditions in the depth of the room have improved.

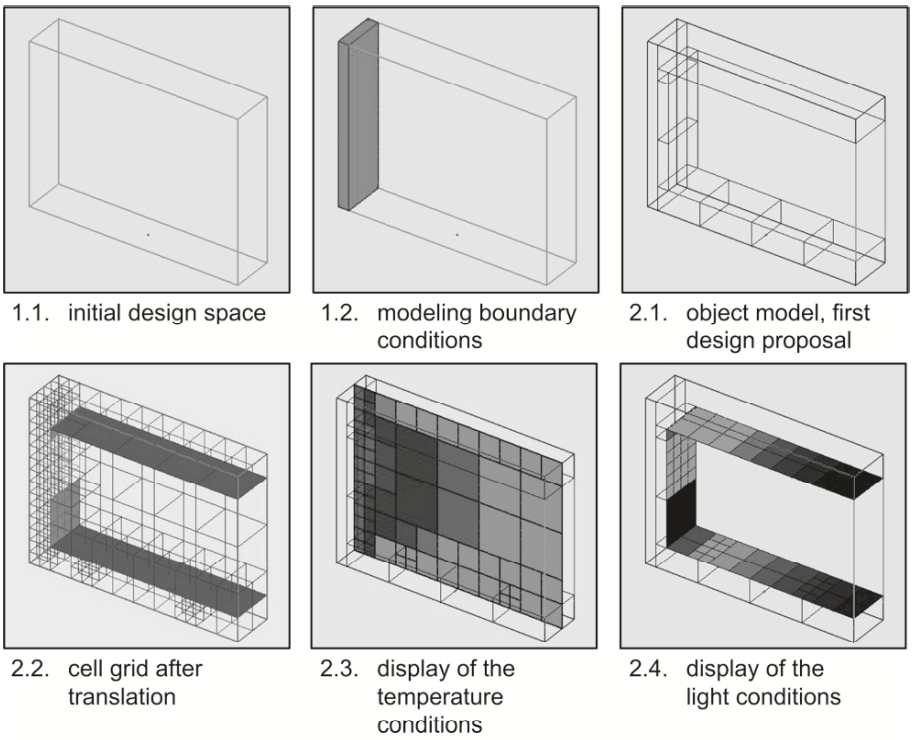


Figure 14. Sequence of steps in the Design Process.

The dark areas in the temperature display, which indicate uncomfortable hot conditions, around the heating element and the light fittings suggest that lower temperatures and control functions for the heating system and the lights are required to prevent the room from over-heating. Further steps of design improvement would follow.

5. Restrictions and Observed Problems

The prototype, as implemented, is currently not applicable in a real world design process. This is due to long processing times for the model translation and the simulation itself. Therefore a constructive dialog between the designer and the simulation tool cannot be established in the design development process.

Furthermore the attempts to implement the flow model, which would be essential for the assessment of comfort quality, were not successful at this stage and other physical models will require further refinement, adjustment and testing.

The size of the model is limited by memory constrains and the resulting translation and simulation time. Small cell sizes (<5cm) occasionally result

in numerical problems during the simulation. Due to the cubic form of the cells a large number of cells and cell faces is generated when models with thin objects (e.g. table tops) are translated. This results in a larger of conjunctions, especially remote-conjunctions, and causes memory problems and long translation and simulation times without adding much accuracy to the simulation results.

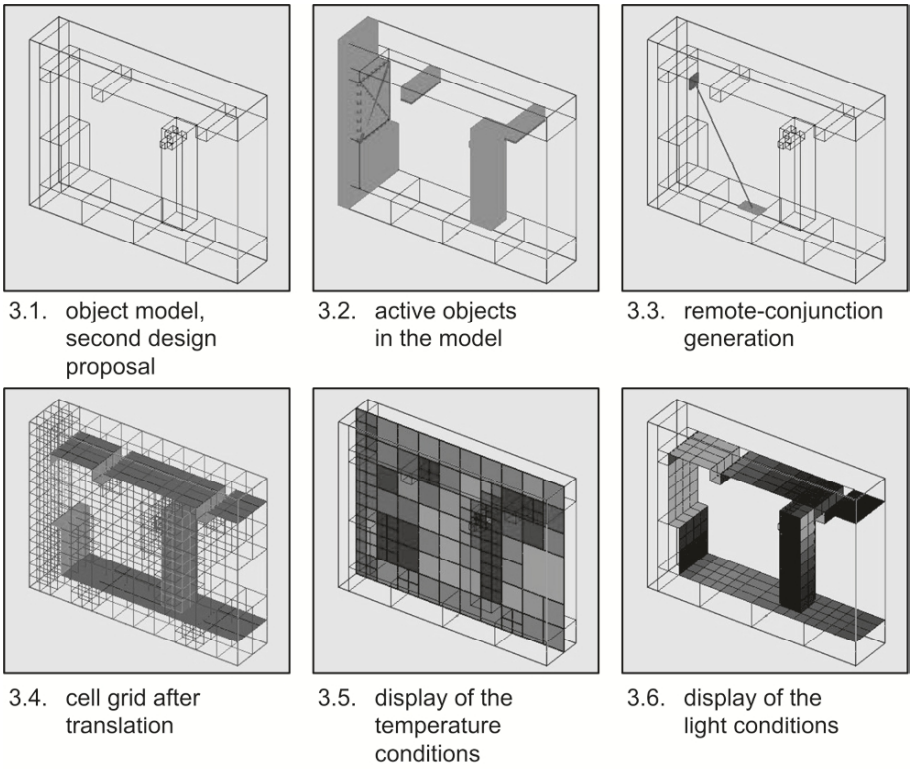


Figure 15. Sequence of steps in the Design Process.

For simplification and robustness of the implementation of the model translation algorithm only rectangular objects with sides parallel to the cell grid can be used for modeling. Nevertheless the concept of the cells and the cells faces would allow the representation of objects with non-rectangular shapes. The accuracy of the translation of such objects would be improved with reduced cell sizes (similar to higher accuracy of pixel graphics with smaller pixel sizes).

Although various problems and limitations exist, the research demonstrates the separate steps in the process and the application of the sequence of steps in the design process successfully. Furthermore the

implemented system shows a way towards highly integrated simulation of physical processes in the digital world.

6. Conclusion

A space-based model to simulate and to display highly integrated physical phenomena on basis of digital object-oriented design representations, which only contain geometry and basic geometry-related property information, is described and demonstrated briefly. Its potential application in the performance-based design process is presented and its current stage of development is discussed.

It must be mentioned that the presented prototype was not developed to be applicable in a real world design process, that technological constrains prevent the modeling of larger model sizes and that the physical model requires further development, refinement and testing, before the system is applicable to predict physical behavior in a useful manner. Nevertheless it was demonstrated how the system would be applicable to bridge complexity of the analysis activity in the digital design process.

Although the focus of this paper is the augmentation of design representations to enable performance-centered designing, further applications could be imagined. For example the concept could be used in physically realistic virtual environments to enable research on overall comfort or research on adaptive comfort models with human-like agents in virtual environments in future.

Acknowledgements

The reported research is supervised by Bruce Forwood (University of Sydney) and supported by the International Postgraduate Research Scholarship funded by the Australian Department of Education, Science and Training (DEST), the International Postgraduate Award (IPA) by the University of Sydney and the Faculty of Architecture of the University of Sydney.

References

ESRU: 2002, *The ESP-r System for Building Energy Simulation - ESRU Manual U02/1*, University of Strathclyde, Glasgow.

Fredkin, E: 1992, Finite nature, Available Online: http://www.digitalphilosophy.org/download_documents/finite_nature.pdf

Gero, JS: 1990, Design prototypes: A knowledge representation schema for design, *AI Magazine* **11**(4): 26-36.

Künzel, HM: 1994, Verfahren zur ein- und zweidimensionalen Berechnung des gekoppelten Wärme- und Feuchteverhaltens *Bauteilen mit einfachen Kennwerten*. Fakultät Bauingenieur- und Vermessungswesen der Universität Stuttgart, Lehrstuhl Konstruktive Bauphysik. Stuttgart, Universität Stuttgart, Germany.

Mahdavi, A: 1999, A comprehensive computational environment for performance based reasoning in building design and evaluation, *Automation in Construction* **8**: 427-435.

Marsh, AJ: 1997, *Performance Analysis and Conceptual Design*, PhD Thesis, School of Architecture and Fine Arts. Sydney, University of Western Australia.

Schwede, DA: 2006, *Towards a Universal and Integrated Digital Representation of Physical Phenomena*, PhD Thesis, Faculty of Architecture, University of Sydney.

Suter, G and Mahdavi, A: 2003, Aspects of a representation framework for performance-based design, *IBPSA conference: Building Simulation 2003*, Eindhoven, Netherlands, Organizing Committee Building Simulation, pp. 1257-1265.

Vischer, J: 1989, *Environmental Quality in Offices*, New York, Van Nostrand Reinhold.

Zuse, K: 1970, *Calculating Space*, Cambridge, MIT Technical Translation of: Zuse, K. (1969) ”Rechnender Raum”, Schriften zur Datenverarbeitung, Vieweg&Sohn, Braunschweig; AZT-70-164_GEMIT, Massachusetts Institute of Technology.

BUILDING CONNECTIVITY MODELS IN DESIGN: REPRESENTATIONS AND TOOLS TO SUPPORT COGNITIVE PREFERENCES

RENÉ KELLER, CLAUDIA M ECKERT AND P JOHN CLARKSON
University of Cambridge, United Kingdom

Abstract. Reasoning about the connectivity within a product is an integral part of many core design activities. Due to the complexity of a product and the sheer number of potential links, designers often overlook vital connections resulting in problems later in the process, leading to errors or costly rework. Product connectivity models, which are essentially graphs, are a promising approach for capturing these links between components in a complex product. The primary visual representation used to create such connectivity models is the Design Structure Matrix (DSM). However, other representations of graphs may be superior for creating connectivity models of products. This paper presents node-link displays as equally valid representations for product connectivity models and reports on an experimental study that investigates whether DSMs or node-link diagrams are more suitable for building such models.

1. Introduction

Models of products and processes play an integral part in reasoning about engineering design. The final product does not exist during the design process in a physical form until a prototype can be made. Even when a physical reference design does exist, designers don't always have access to it, for example when it is too large (Boeing 747 jet) or too small (microchip); the behaviour and function of an object is not immediately visible from its physical form.

Process models and plans, on the other hand, are the only representation of the design process. However, they are notoriously inaccurate and ambiguous regardless of which of their three potential roles they play at any one time: as monitoring aids; prescriptions of the process; or records of the process (Eckert and Clarkson 2003). For very complex products or processes, it is impossible for one person to remember and recall all necessary (detailed) information. This makes models of products and

J.S. Gero (ed.), Design Computing and Cognition '06, 41–60.
© 2006 *Springer. Printed in the Netherlands.*

processes vital as externalisations of collectively-held knowledge that enable individuals to reason about that knowledge and for teams to communicate. Thus it is very important for the success of any design project that every member of the design team has a consistent understanding of the models they use (Henderson 1999).

However, each product or process has multiple models, which might overlap, and which are subject to different interpretations depending on the person and the time in the design process. Some models are used within one group, with a joined understanding; some are boundary objects and serve as communication aids between groups. While reading models is, to some extent, an acquired skill, these boundary objects need to be easily understandable to be accessible. This paper talks about boundary type objects (Star 1989) that bridge the gap between groups with different expertise while expressing very complex information. Thus, these models must be fairly abstract to capture complex information in a concise form, but also fairly rich to provide a meaningful range of connections.

As models are often the only reference to the process or the product, understanding and interacting with these models is an integral part of the design activity. Through design processes, models can be used in very different ways and have different functions.

- Cognition: Human beings are severely limited in the complexity of the things they can keep in mind at one time (Cowan 2001). As Simon (1998) pointed out, designed complex systems are organized as 'nearly decomposable' hierarchical structures with components whose interactions are much simpler than their internal workings, so that it is feasible to understand each element in terms of its behaviour and the interactions of its subcomponents. For large and complex products and processes, models can be a means of abstracting knowledge to break it down into an understandable and manageable piece of information of medium size (Zeitz 1997). An alternative, and complementary, approach to reducing the complexity of the thinking designers need to do is to consider different *aspects* of a design separately. For instance, Hoover et al. (1991) observed designers employing different abstractions and corresponding graphic representations to perform analyses of different aspects of their designs. By "blending out" unnecessary information, such as using fisheye approaches (Furnas 1986) and grouping similar things together, they can allow effective information retrieval even for complex products and processes, and form the basis for analyses of certain aspects of the product or process.
- Communication: One important application of models is to communicate information to other stakeholders. Designers carry models to other people's desks in order to clarify certain aspects of a design, and bring models to meetings. Especially when people have different backgrounds, hence a different knowledge of the design, which is

inevitable the case in the design of very complex products (Jarratt et al. 2004b), suitable models can be seen as a "least common denominator" that is understood by the whole design team. Models are also a way of communicating ideas about the design to people outside the design team, or to managers who only have a very broad overview rather than a detailed technical understanding of the process or product.

- Recording: Models can be used to store information for further designs. Models of previous design processes and products, for instance, can be used as a starting point for new designs and are also a way to train novice designers. However, when using models there is always a trade-off between the amount of effort it requires to create and update the model and the expected value it can provide.

Models (not only process and product models) also have several different possible visual representations, e.g. connectivity models can be represented for instance as node-edge diagrams or matrices. The chosen representation can have a huge impact on how people interact with and interpret information provided. Each type of presentation has a structure and only "affords" the representation of certain type of information, as "*there is no one representation that allows detailed considerations of all possible concerns*" (Gero and Reffat 2001). Larkin and Simon (1987) highlight the importance of a proper visual representation and state that "*whether a diagram (or any other representation) is worth 10,000 words depends on what productions are available for searching the data structure, for recognizing relevant information and for drawing inferences from that information.*"

Thus, a proper representation must allow the user to perform the desired task on the underlying model easily. Many representations of a product or a process must serve very different purposes and should support different designers in all stages of the design process. They should provide a means to communicate about the underlying thing (the product or process). The representations should also support users when building the model in the first place, thus reducing the effort required for creating and maintaining the model. However, this paper will show that how easily information can be extracted from a representation, depends highly on the knowledge of the person interacting with the model and their personal preference.

This paper discusses which representation of a particular kind of model (connectivity model) is generally better suited for the particular task of building such a model. In order to achieve this, experiments were undertaken that compared the two most common representations for connectivity models, namely node-link diagrams and matrix-based representations, in the context of product connectivity models. This research completes the contribution on representations of connectivity models, where in a second

study (Keller et al. 2005a), the differences between representations for information retrieval tasks were analysed.

The remainder of this paper includes:

- A description of current connectivity models of products and processes and their applications in industry (Sections 2);
- An introduction and comparison of Design Structure Matrices and node-link diagrams used to represent product connectivity models of complex products (Section 3);
- Presentation and discussion of an experimental user study that shows how the representation (matrix-based or node-link) influences connectivity model-building (Section 4).

Understanding the implications of such a comparison may be very beneficial to the design process as well as for other disciplines where such connectivity models are common (e.g. social networks in social psychology).

2. Connectivity Models

As in any complex, almost decomposable system, linkages between parts are a common structural feature in both processes (dependencies between tasks) and products (e.g. components that have spatial relations). Connectivity models capture those connections and can be used by designers and project managers to model products as well as processes and projects. For example, task dependencies in process models can indicate the order of tasks in a process. The main underlying structure is a graph, with nodes and links modelling interactions between nodes. A famous process model of this kind is the PERT method (Malcolm et al. 1959), which is used to find critical paths in the tasks that constitute a project. Design Structure Matrices (DSMs) are a method to represent and restructure dependencies of the binary representation of a matrix-based representation. For example, DSMs are used to improve the design process by reordering the task sequence (Browning 2001). Signposting models of design processes (Clarkson and Hamilton 2000) is a generalization of the relational model of task-dependencies, where dependencies are driven by the state of parameter descriptions.

2.1. PRODUCT CONNECTIVITY MODELS

The range of general product models includes detailed CAD models of geometric and electrical properties, functional models, component breakdowns of products (such as the bill of material), and prototypes and sketches (Henderson 1999). All represent information about the product at different levels of abstraction and from different viewpoints. CAD models, which use a very concrete representation of the spatial relations of components of a product, are widely used in industry (see (Henderson 1999)

for the description of case studies revealing the role of CAD models in engineering companies) and play an integral role in the design of new products. More abstract product models like functional models (Pahl and Beitz 1996) are used in conceptual design. However, currently very few models indicate relations between parts and combine these different aspects of the product into one model or even a coherent set of models.

A product connectivity model captures the components of a complex product and the different interactions between its parts in an abstract way as a graph. The possible relations between components of a product can be manifold and depend on the particular application of the model. In recent case studies, an extensive list of different linkage types was used (Jarratt et al. 2004a) to take this into account. The list groups individual parameters to provide an abstract yet specific view.

A challenge for product connectivity models is building models for large and complex products. Products such as helicopters or gas turbines consist of several thousand components, connected in various ways, designed by multidisciplinary teams. Such models need to be hierarchical, representing different levels of abstraction. The models in this paper are small and represent component breakdowns with a size of not more than 100 components. These are usually, again, an abstraction of a more complex model. Thus it is vital that a proper component breakdown is established before the actual model-building exercise.

2.2. PRODUCT CONNECTIVITY MODELS FOR CHANGE PREDICTION

One application of product connectivity models is change prediction (Clarkson et al. 2004). The change prediction method computes risks associated with component changes once the design of a product is finished and allows designers to foresee effects of changes to components before these changes are implemented.

The product connectivity models used for change prediction are elicited in a group meeting with experienced designers who all have different views and knowledge of the product, due to their different backgrounds and responsibilities. The main representation in this model-building exercise is a Design Structure Matrix. Such a model-building session with a group of designers consists of four steps:

1. A component breakdown of a medium level of abstraction is established;
2. A list of possible linkage types is created;
3. The group methodically goes through the list of all pairs of components and decides whether there is a link between these two components and of which type this link is;

4. Direct change likelihood and impact values are assigned to each of the established component connections.

With this methodology it is hoped that most of the links (even very subtle ones) can be detected, while balancing individual biases. As usually very few designers have a complete overview over the entire product (Jarratt et al. 2004b), designers responsible for different aspects of the design are invited to such a meeting to contribute their rows and columns. The values for change impact and likelihood and links between the components elicited in this way reflect the experience of these designers and are usually based on previous designs. However, as Ayton and Pascoe (1995) point out, it is questionable whether these change values truly affect real-world change propagation probabilities, because people generally make mistakes when judging uncertainties.

The benefits of creating a product connectivity model for change prediction are three-fold.

- Learning in the group: A group of designers collectively build the model (Jarratt et al. 2004b) where many of the 'war stories' of the design process emerge. Each designer puts in his or her knowledge and throughout the exercise all designers gain knowledge, even on parts of the final product they are not directly involved with. The exercise also helps to reveal the need for interaction between different design teams.
- Model for analysis: The information on direct change likelihood and impact values assigned to component connections alone can reveal risky component connections. They can also be used to calculate indirect change risks (Clarkson et al. 2004) using the change prediction method and thus to predict the impacts and risks associated with changes to components resulting from indirect connections.
- Product overview: In design decision-making, product connectivity models can provide a necessary overview of the product that can be updated during the product lifecycle. It can even provide a way to integrate new members into the design team, by providing the information stored in the model.

2.3. INDUSTRIAL RELEVANCE OF PRODUCT CONNECTIVITY MODELS

Product connectivity models for change prediction proved their value in several case studies carried out by members of our group. The range of companies involved includes a helicopter company (Eckert et al. 2004), a diesel engine manufacturer (Jarratt et al. 2004a) and a UK aerospace company. Throughout these case studies the change prediction method and a corresponding software support tool (CPM tool) for analysing change propagation were developed and applied. The industrial success of such a change propagation tool, however, also depends on finding a way to present

all the desired information visually in such a way that the user (in this case the designer) is not overwhelmed by the amount of information.

The feedback from all the companies that used the product connectivity models was generally positive. The models were used for very different applications, ranging from risk assessment of component change to storing important information on the product. All companies agreed that especially the model-building exercise had a positive impact on the design team when designers involved in the process came together and everyone added his knowledge to the model. We are currently looking for means to improve the CPM tool and especially its human–computer interfaces for building such connectivity models, which is the main motivation for the comparison between DSMs and node-link representation for model-building introduced in the remainder of this paper.

3. Visual Representations of Connectivity Models: DSMs and Node-Link Diagrams

As mentioned earlier, graph structures - the basis for connectivity models - can be represented in different ways. Most common are matrix-based representations and node-link displays. Both representations are equally valid representations of graphs. This section will introduce these two most common visual representations for relational data. It will also show how both representations support the visual display of product connectivity models. These visual representations not only show information provided by the model once it is built, but can also be used to build the model in the first place, which will be the main focus in this paper.

3.1. DESIGN STRUCTURE MATRICES

A Design Structure Matrix, (Steward 1981) is presented as "*a simple, compact, and visual representation*" (Browning 2001) in various literature sources. However, a DSM is not the primary representation that designers would use. For example, a design manager from a UK gas-turbine company responded to the question of whether he used a DSM or a node-link representation when he created a functional model of a gas turbine that he "*created a network* (node-link diagram) *first and then transformed it into a DSM*". Another designer had more general reservations, saying: "*Lets face it, a DSM is not a representation designers like using*". In another user-study conducted on the different factors that influence model-building with DSMs an experienced designer with dyslexia had particular problems building a product model with a DSM and was keen to use a node-link representation. This leads to the question whether a DSM is the most appropriate representation for product model-building as used in current methodologies.

DSMs are essentially adjacency matrices and thus squared matrices; each node of the underlying graph is represented by a row and a column. A mark in a DSM means that there is a link from the element represented by the column of the matrix to the component represented by the row. The definition of how to read such matrices varies amongst different research communities, as it is not inherent in the matrix how it should be read (in this paper, a mark always represents a link from the column to the row). Figure 1 shows two resulting DSMs of the core components of a simple car engine example based on different component orders. These DSMs show only mechanical links.

Simple Engine		1	2	3	4	5	6
Camshaft	1	1			X		X
Crankshaft	2		2	X		X	
Cylinder Block	3		X	3	X	X	
Cylinder Head	4	X		X	4		X
Piston	5		X	X		5	
Valves	6	X			X		6

a)

Simple Engine		6	1	4	3	2	5
Valves	6	6	X	X			
Camshaft	1	X	1	X			
Cylinder Head	4	X	X	4	X		
Cylinder Block	3			X	3	X	X
Piston	5				X	2	X
Crankshaft	2				X	X	5

b)

Figure 1. Two DSMs of a simple car engine, a) alphabetical order, b) ordered by location in the engine. Each mark in the matrix represents a link from the column-component to the row-component.

The advantage of matrix-based representations is that the possible number of different layouts for such a DSM is restricted to the order of the elements in horizontal and vertical direction. Anecdotal evidence indicates that once an order for the components is established, subjects find it especially easy to find a component in a DSM they have seen before. This doesn't seem to be influenced by what the original order of the components is as long as it is maintained during the entire lifetime of the model so that subjects are familiar with the order. These findings correspond with research into the order of menu items (Card 1981). Somberg (1987) for instance found that users performed fastest with a fixed menu item order when users had experience with the position of the items. Typically, elements of a DSM are ordered by importance, either by their arrangement in the product or by alphabetical order.

However, changing the order of the elements of the matrix can be beneficial for further analyses of a DSM. Techniques such as sequencing or clustering (Browning 2001) of the matrix change the given order to show aspects of the data that cannot be easily examined within the original view (see Figure 1 for two different orders of the same DSM). The advantages of this reordering can be manifold. Sequencing allows the establishment of an

order in which (process) tasks have to be executed. Clustering techniques can identify highly-connected clusters of components, which can be the basis for a component breakdown of the entire product.

3.2. NODE-LINK DIAGRAMS

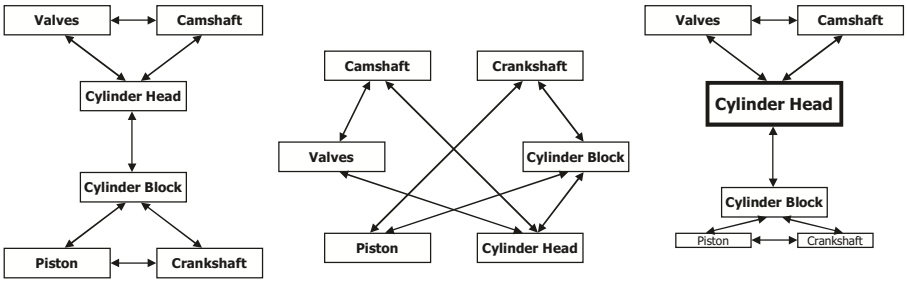
A node-link representation of a connectivity model potentially carries the same information and can be easily transformed into a DSM and vice versa. This begs the question whether it is as useful for representing connectivity models in general, as is a DSM. We think that designers and other users who need to interact with connectivity models should use the representation that is best suited to their particular needs, when building or interacting with models. The main contribution of this paper is the comparison between a DSM and a node-link representation for connectivity model-building. How useful DSMs and node-link diagrams are for analysing and showing such data will not be addressed here and is an area of ongoing research (Keller et al. 2005a). A comparison between the visual affordances of graphs and matrices for reading important information from the representation of this kind of data can also be found in Ghoniem et al. (2004).

In a node-link diagram, each component of a product or process connectivity model is represented as a node, and edges between nodes represent links between these components (see Figure 2 for node-link representations of the car engine model also shown in Figure 1 with a matrix-based representation). For the layout of such a node-link diagram, the entire two-dimensional space can be used. Thus, the number of layouts is much larger than is possible with matrix-based representations. This larger variety of possible layouts allows displays to focus on different aspects of the data. These include:

- Spring Layouts (Huang et al. 1998) to show clusters.
- Hierarchical networks (Schaffer et al. 1993) that can visualize the component hierarchy of products for instance.
- Fisheye views (Furnas 1986) and radial layouts (Jankun-Kelly and Ma 2003) for focusing on one node or a group of nodes.

An extensive collection of possible layout algorithms for node-link diagrams can be found in di Battista et al. (1994). See Figure 2 for some examples of different layouts for a simple graph.

Displaying relational data in a node-link representation, however, has some disadvantages and problems. Different layouts, for instance, give room for ambiguity and especially for very large graphs, the problems of edge-crossings and overlapping nodes can be very severe. While small and sparse node-link diagrams can often be drawn without edge-crossings (these graphs are then called planar), especially large graphs and graphs that are highly connected (with many links between components) cannot be laid out properly.



(a) (b) (c)

Figure 2. Three node-link diagrams of a simple car engine, a) planar layout without edge-crossings, b) circular layout, c) fisheye view of the graph with a focus on the 'Cylinder Head'.

See Ghoniem et al. (2004) for the implications of size (number of nodes in the underlying graph) and density (number of links divided by the possible number of links) for the readability of node-edge diagrams and matrix-based representations. Unfortunately, some product models are very complex: very large (a helicopter for example has more than 10,000 distinct parts) or very dense (in a diesel engine model, we found that each single component is on average connected to 6 other components which gives a density of 28%). In that case it is vital (also if a matrix-based representation is used) to find an appropriate level of abstraction.

3.3. IMPLICATIONS OF DIFFERENT REPRESENTATIONS FOR PRODUCT CONNECTIVITY MODELS

The main representation implemented in current state-of-the-art software for building connectivity models is the Design Structure Matrix. One example of such a program is the CPM tool described earlier (Jarratt et al. 2004a) for engineering change prediction, which uses a DSM as the main representation for model-building as well as for further analyses of change propagation paths.

Observations of product connectivity model-building in industrial case studies have shown that there is a gap between current methodologies and software support tools on the one hand and what designers seem to prefer on the other hand. While computer support tools, such as the CPM tool, usually incorporate DSMs as the primary representation, we found that the preferred representation depends highly on the designer and the task. In this paper we investigate which representation is more suitable for connectivity model-building using both qualitative (case study observations of using DSMs and node-link diagrams for model-building) and quantitative (an experiment that tests user performance with both techniques) methods. The goal is to include

this representation in a computer tool that supports model-building of product connectivity models.

4. Experimental Study

In order to quantify whether a DSM or a node-link representation is better for product connectivity model-building, we conducted a psychological user-experiment (Martin 2003). The aim was to reveal differences in user performance using the two different representations.

For this experiment, 27 participants, all engineering students, ranging from first year students to PhD candidates with practical design experience and PostDocs, were recruited. In a video-recorded pre-study consisting of individual sessions with 6 participants, the general layout of the study was evaluated. Subsequently we held the experiment with the remaining 21 participants. The participants were paid £10 for their time.

Each participant was given sketches of two products, a drawing of the human heart consisting of 8 components and a drawing of a car engine with 16 components, Figure 3. Both examples represented simple systems, with which most of the participants were familiar. This is also true for the heart model, which does not originate from the engineering domain, but due to its simplicity, we believed that is represented a similar level of familiarity to the participants. Time constraints prevented us from using larger models as for instance the diesel engine model that was used in previous studies (Jarratt et al. 2004a).

The given component breakdown was necessary in order to be able to compare different results and also represented a medium level of abstraction. The participants were asked to complete the two models sequentially in the order they were given them. However, we observed one participant working on both examples simultaneously, and not following this order (this is the outlier in the completion time dataset discussed below in Section 5.1.1).

In this study we were interested in how to model spatial relationships, as these are the easiest to infer from a drawing. The participants were asked to create a connectivity model of each of the products, where the linkages should reflect mechanical or spatial relationships between components of the product. We were not interested in the thermal or fluid flow relationships, as we believed that only experts could assess most of these additional links properly.

Each participant then had to complete one model using a DSM and the other one using the node-link representation (so half of the participants completed the heart with a DSM, the other half with a node-link diagram). Additionally, we asked for their experiences with DSMs and their current level of study. No tools other than a pen and paper were allowed. There was no time limit so all the participants had as much time as they needed. The

total time spent by the participants including an introduction phase was about 50 minutes with a maximum of 1 hour.

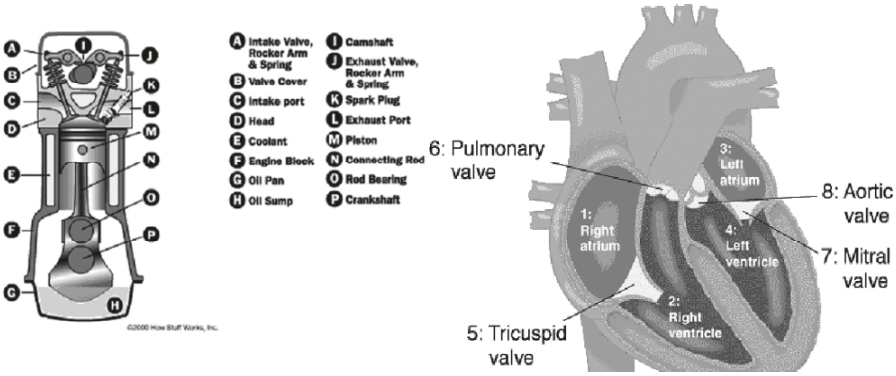


Figure 3. Example 'products' used in the experimental study: An engine with 16 components and a heart with 8 'components'.

Response times were recorded, as well as the resulting node-link diagrams and DSMs. In order to create a single representation, and for calculation issues, all resulting models were transformed into DSMs and then analysed. A link between two components was coded as 1, no link as 0. The results of one user were removed from the dataset as he filled out the DSM for the car engine with a regular pattern.

4.1. COMPARISON BETWEEN DSMS AND NODE-LINK DIAGRAMS

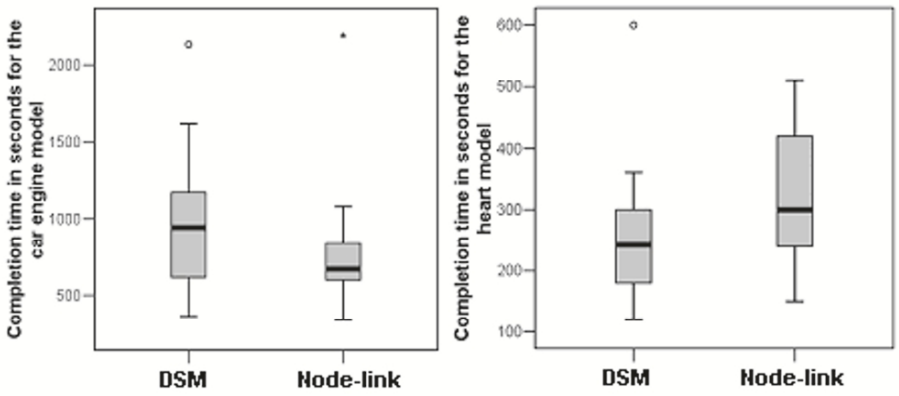
In this section we show the results of the experimental study for the comparison between a DSM and node-link representation. We focus on three different variables, namely the completion time, the number of links found, and the variations found amongst different solutions. We were particularly interested in the effect of the visual representation on these variables and not so much in the correctness of the answers given, as we did not expect anyone to have a very detailed understanding of a car engine or a human heart. For the analysis of the results we used parametric statistical tests as well as a qualitative graphic method (box plots, see Chambers et al. (1983)). Box plots are a well-known way of representing the density of data by showing important statistics (the box of the plots for instance shows 50% of the data, the central horizontal line represents the median).

4.1.1. Completion Time

Initially we were interested in the differences in the completion times for the entire product model. Completion time is a standard measure used in several other studies when comparing readability of node-link diagrams and matrix-based representations (Ghoniem et al. 2004). The null-hypothesis H_0 that

there are no differences between the two groups could not be rejected for both models, the car engine and the human heart (using a t-test, as we consider response times as normal-distributed). This means that the differences were not statistically significant.

However, the box plot for the car engine in Figure 4 shows that there is one strong outlier in the response times for the large product model of the car engine. One participant needed significantly more time than any other participant filling out the node-link diagram (see the explanation earlier, he did both models simultaneously). A Grubbs's outlier test also showed that this value is an outlier with a probability of more than 99%. Without this outlier the completion time for the node-link representation was significantly (with a significance level of α=5%) shorter compared to the completion time needed for a DSM. In the model of the heart, even removing the outlier shown in Figure 4(b) does not change the fact that there are no significant differences in the completion times, although in that example, the completion time of a DSM was shorter than that of a node-link representation.



(a) (b)

Figure 4. Box plots of the completion time for the car engine model (a) and the heart model (b).

4.1.2. Number of Links

Secondly, we were interested in the number of links found by participants using DSMs and node-link diagrams. The number of links shows whether subjects really considered every possible connection between components and found even links that are not obvious in the first place. We argue that the more links that were found, the more attention was paid to even very weak links and thus, the more complete the model.

We found that the number of links found by subjects using a DSM for the car engine model was significantly larger (α=10%) than the number of links found with a node-link diagram using a t-test. The number of links is binomial-distributed, and as n is large (n=56 or n=240 possible links in the models), the total number of links can be modelled as normal-distributed. Generally, more links were found with a DSM. See Figure 5 for the corresponding Box plots that support this thesis. For the heart model again, no significant differences were detectable. However, it can be seen in the corresponding Box plot, Figure 5, that there is a higher number of links found with the DSM representation.

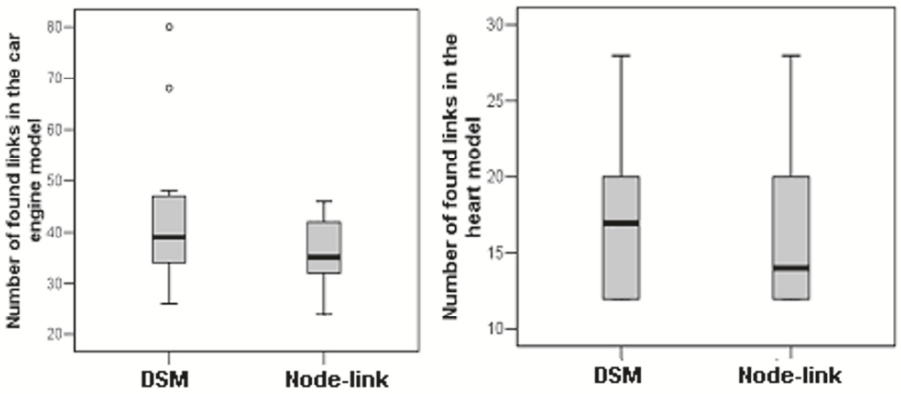


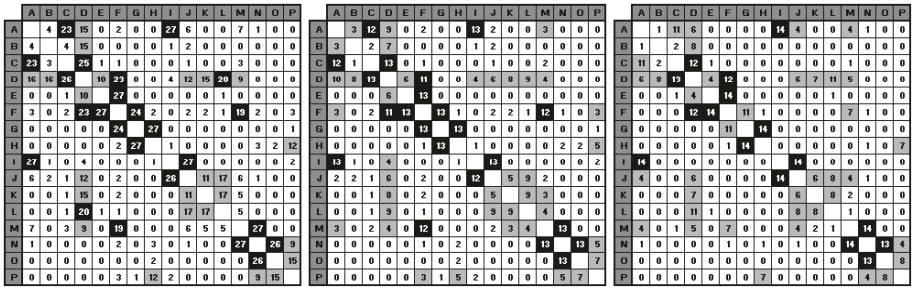
Figure 5. Box plots of the number of found links for the car engine model (a) and the heart model (b).

4.1.3. Variance

In this section we will analyse how the representation of the connectivity model influences the variation of solutions qualitatively. Figure 6 shows the results graphically. On the left there is the DSM that incorporates all 27 solutions (by the 27 participants) of the car engine (the node-link diagrams were transformed into DSMs and then all these DSMs were added). The DSM in the centre shows the solutions that were created by participants using only the DSM, the one on the right shows the results of the users using the node-link representation.

As one can easily see, in general, there is a lot of variation between different solutions. The colour coding in the cells represents whether the number of solutions that had this link is significantly ($p<0.05$ under a binomial distribution) smaller than 0.5 (white background), significantly greater than 0.5 (black background) or not significantly different from 0.5

(grey). The cells with a grey background represent links where there was a high controversy amongst the different solutions in a group.



(a) (b) (c)

Figure 6. DSMs showing all solutions, the DSM solutions and the node-link diagram solutions.

However, as Figure 6 shows, there are hardly any differences between the DSM and node-link diagram solutions. The only real difference is the link between components M (Piston) and F (engine Block), which was found by almost all participants using the DSM (12 out of 13) but was only found by roughly half of the participants using the node-link representation (7 out of 14).

Additionally, one can see that there are more black boxes (i.e. cells with a link that is significantly bigger than 0.5) amongst the DSM solutions (22) than amongst the node-link diagram solutions (16). This corresponds with the finding that, generally, more links were considered using a matrix representation.

4.2. SUMMARY

The experimental study showed that the difference between a DSM and a node-link representation small when used to build product models in an engineering context. We found that the participants using a DSM assessed more links than those using a node-link representation. The time spend on the node-link representation, however, was shorter (for the car engine model). This might be the result of the smaller number of links considered with the node-link diagrams, and for the smaller heart model, participants were faster using DSMs. After the study, we also asked which representation each subject favoured. The answers were even (13 liked DSMs more and 13 node-link diagrams, while one participant could not decide), so there were no detectable user preferences.

5. Discussion

The study introduced in the previous section showed that the differences in user performance using DSMs and node-link representations for product model-building are small. It could be observed that subjects were faster with node-link diagrams (at least for larger product models, such as the engine), and that, using DSMs, in general more links were considered. However, the combined solutions - as seen in Figure 6 - do not significantly differ, but individuals seem to have strong opinions on when to use which representation. Other factors, such as experience of the participants, however, can have huge impacts on how subjects perform at a product model-building exercise, but were not considered in this study.

As stated earlier we were looking for the "best" possible representation for building product connectivity models so that it could be incorporated into the CPM tool. The study showed that the differences between the two representations are small and we see that there seems to be no best representation for building product models. However, it was clearly shown that participants using DSMs filled in more links than those using the node-link representation. Especially in the context of change prediction and propagation, where hidden links can cause huge problems, this is a strong argument for using DSMs for model-building rather than node-link diagrams. The structured way of filling out a DSM (people responded that they usually went column or row-wise through the matrix) lets them consider even links that are not obvious in the first place.

This and the fact that, on the other hand, some users have problems using matrix-based techniques lets us propose that the users should be able to decide which representation suits them best for model-building and thus, as an implication for future software tools, both representations should be incorporated into a software tool in order to effectively support designers in building connectivity models in general. Although this approach means more work for the programmer responsible for the implementation of adequate user interfaces, the benefits for users can be very high and almost no user will be excluded, as opposed to current approaches, which just provide a DSM. The requirements for such a tool must also include a means of linking both representations. Changing the underlying model in one representation should have immediate effects on the other representation. This concept of linking different graphs and representations is especially common in the field of interactive statistical graphics (Unwin 1999). It also follows that, especially when building product models in a group meeting, computer tools are necessary, because it might be beneficial to swap between different representational modes during the meeting to benefit from different representations of the product model.

The CPM tool for predicting change propagation in complex products is capable of supporting the model-building side of the design process. However, while multiple representations are already used for analysing and presenting change data and the resulting matrices (Keller et al. 2005b), the model-building capabilities of the software are still limited to using only DSMs as the primary representation. Our final goal is to bring an updated version of the tool back into an industrial model-building meeting.

Another implication for the use of product models in engineering design is that one cannot rely on models created by one person. As the experimental study showed, there is a lot of variation between different solutions regardless of the visual representation. Future research should reveal whether the current strategy of model-building (a group meeting) is an appropriate methodology. This evidence however supports the current methodology of having a group meeting with different designers rather than relying on a single opinion. The differences among the results detected in this study (where all participants shared a common understanding of the product) will be even greater in an industrial setting where every designer is responsible for a different part of the design and has a different (academic) background.

Furthermore, the implications from this study can be generalized to all sorts of representations that rely on graph structures, such as process models. Building process models involves similar strategies; the structured way of creating models using a matrix-based representation can be very beneficial as even hidden links can be detected more easily.

6. Conclusions

In this paper we compared the visual affordances of DSMs and node-link representations and their ability to support construction of product connectivity models. We found that DSMs and node-link representations can both benefit product model-building differently, as they propagate different building strategies. A supporting experimental study for connectivity model building was conducted, which indicated that DSMs are slightly better for assessing linkages between components of a complex product as participants considered more links when using a DSM than using a node-link representation. It was argued that this is due to the more structured way of representing the data using matrix-based techniques. This finding did not correspond to anecdotal evidence from designers in industry who preferred node-link representations for building connectivity models. However, the overall differences between subjects using different representations were relatively small due to individual preferences. This study also rounds up the comparison of matrix-based representations and node-link representation for connectivity models. As shown in previous studies (Ghoniem et al. 2004;

Keller et al. 2005a), node-link representations are better for reading information from small and sparse graphs and are generally better suited for showing information about indirect links between two components, while matrix-based representations seem to outperform node-edge diagrams when a more structured way of interaction with the underlying model is required, such as was shown here for product connectivity model-building.

Due to the small differences, we propose a strategy for connectivity model-building that makes use of multiple representations rather than one primary visual representation. This would allow designers in industry to choose their preferred representation, and would benefit the design process by facilitating the integration of their knowledge into the product models. The differences between individual solutions suggest that it would be preferable to rely on multiple opinions than on the judgments of one single expert, which is supported by current methodologies (Jarratt et al. 2004a; Austin et al. 2001). However, whether this knowledge should be elicited in a meeting with different designers or whether several solutions by single designers should be incorporated into one model by a researcher is still unresolved and is a topic of future research.

The results of this paper mainly refer to building product connectivity models in an engineering context. However, as connectivity models of processes (as for instance task DSMs or PERT methods) follow similar concepts in terms of connectivity and the creation of such models is also highly dependent on the visual representation of the underlying graph structure, these findings should be also applicable for building such models.

Acknowledgements

This research is funded by EPSRC.

References

Austin, S, Steele, J, Macmillan, S, Kirby, P and Spence, R: 2001, Mapping the conceptual design activity of interdisciplinary teams, *Design Studies* **22**(3): 211-232.

Ayton, P and Pascoe, E: 1995, Bias in human judgement under uncertainty?, *The Knowledge Engineering Review* **10**(1): 21-41.

Browning, TR: 2001, Applying the design structure matrix to system decomposition and integration problems: A review and new directions, *IEEE Transactions on Engineering Management* **48**(3): 292-306.

Card, SK (ed): 1981, User perceptual mechanisms in the search of computer command menus, *Proceedings of the 1982 Conference on Human Factors in computing systems*, Gaithersburg, Maryland, US, pp. 190-196.

Chambers, J, Cleveland, W, Kleiner, B and Tukey, P: 1983, *Graphical Methods for Data Analysis*, Wadsworth.

Clarkson, PJ and Hamilton, JR: 2000, 'Signposting', A parameter-driven task-based model of the design process, *Research in Engineering Design* **12**(1): 18-38.

Clarkson, PJ, Simons, C and Eckert, CM: 2004, Predicting Change Propagation in Complex Design, *ASME Journal of Mechanical Design* **126**(5): 765-797.

Cowan, N: 2001, The magical number four in short-term memory: A reconsideration of mental storage capacity, *Behavioral and Brain Sciences* **24**: 87-185.

Di Battista, G, Eades, P, Tamassia, R and Tollis, IG: 1994, Algorithms for drawing graphs: An annotated bibliography, *Computational Geometry: Theory and Applications* **4**(5): 175-198.

Eckert, CM and Clarkson, PJ: 2003, The reality of design process planning, *ICED 03*, Stockholm, Sweden (on CD).

Eckert, CM, Clarkson, PJ and Zanker, W: 2004, Change and customisation in complex engineering domains, *Research in Engineering Design* **15**(1): 1-21.

Furnas, GW: 1986, Generalized fisheye views, *Proceedings of CHI '86*, Boston, Massachusetts, USA, pp. 16-23.

Gero, JS and Reffat, RM: 2001, Multiple representations as a platform for situated learning systems in designing, *Knowledge-Based Systems* **14**: 337-351.

Ghoniem, M, Fekete, J-D and Castagliola, P: 2004, A comparison of the readability of graphs using node-link and matrix-based representations, *Proceedings of InfoVis 2004*, Austin, Texas, USA, pp. 17-24.

Henderson, K: 1999, *On Line and On Paper*, The MIT Press, Cambridge, MA, USA.

Hoover, SP, Rinderle, JR and Finger, S: 1991, Models and abstractions in design, *Design Studies* **12**: 237-245.

Huang, ML, Eades, P and Wang, J: 1998, Online graph drawing using a modified spring algorithm, *Australian Computer Science Comm: Proceedings 21st Australasian Computer Science Conference, ACSC* **20**(1): 17-28.

Jankun-Kelly, TJ and Ma, K-L: 2003, MoireGraphs: Radial focus+context visualization and interaction for graphs with visual nodes, *Proceedings of InfoVis 2003*, Seattle, Washington, USA, IEEE Computer Science Press, pp. 59-66.

Jarratt, T, Eckert, CM and Clarkson, PJ: 2004a, Development of a product model to support engineering change management, *Proceedings of Tools and Methods of Competitive Engineering (TCME)*, Lausanne, Switzerland: 331-342.

Jarratt, T, Eckert, CM, Clarkson, PJ and Stacey, MK: 2004b, Providing an overview during the design of complex products, *in* JS Gero (ed), *Design Computing and Cognition, DCC'04*, Kluwer, Dordrecht, pp. 239-258.

Keller, R, Eckert, CM and Clarkson, PJ: 2005a, Matrices or network diagrams: Which visual representation is better for visualising connectivity models? *CUED/C-EDC/TR137*, Cambridge, University of Cambridge.

Keller, R, Eger, T, Eckert, CM and Clarkson, PJ: 2005b, Visualising change propagation, *Proceedings of the 15th International Conference on Engineering Design (ICED '05)*, Melbourne, Australia (on CD).

Larkin, JH and Simon, HA: 1987, Why a diagram is (sometimes) worth ten thousand words, *Cognitive Science* **11**: 65-99.

Malcolm, DG, Roseboom, JH, Clark, CE and Fazar, W: 1959, Applications of a technique for research and development program evaluation (PERT), *Operations Research* **7**(5): 646-669.

Martin, D: 2003, *Doing Psychology Experiments*, Wadsworth.

Pahl, G and Beitz, W: 1996, *Engineering Design - A Systematic Approach*, London, Springer-Verlag.

Schaffer, D, Zou, Z, Bartram, L, Dill, J, Dubs, S, Greenberg, S and Roseman, M: 1993, Comparing fisheye and full-zoom techniques for navigation of hierarchical clustered networks, *Proceedings of Graphics Interface '93*, Toronto, Canada, Morgan-Kaufman, pp. 87-96.

Simon, HA: 1998, *The Sciences of the Artificial*, Cambridge, MA, MIT Press.

Somberg, BJ: 1987, A comparison of rule-based and positionally constant arrangements of computer menu items, *SIGCHI/GI conference on Human factors in computing systems and graphics interface*, Toronto, Ontario, Canada, pp. 255-260.

Star, SL: 1989, The Structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving, *in* L Gasser and MN Huhns (eds), *Distributed Artificial Intelligence*, Pitman London, UK, pp. 37-54.

Steward, DV: 1981, The Design Structure System: A method for managing the design of complex systems, *IEEE Transactions on Engineering Management* **28**(3): 71-74.

Unwin, AR: 1999, Requirements for interactive graphics software for exploratory data analysis, *Computational Statistics* **14**: 7-22.

Zeitz, CM: 1997, Some concrete advantages of abstraction: How experts' representations facilitate reasoning, *in* PJ Feltovich, KM Ford and RR Hoffman (eds), *Expertise in Context: Human and Machine*, AAAI Press, pp. 43-65.

GEOMETRIC, COGNITIVE AND BEHAVIORAL MODELING OF ENVIRONMENTAL USERS

Integrating an agent-based model and a statistical model into a user model

WEI YAN
Texas A&M University, USA

and

YEHUDA E KALAY
University of California-Berkeley, USA

Abstract. This paper describes our user model (Virtual User) for behavior simulation. The model simulates the goals, social traits, perception, and physical behaviors of users in built environments. It includes three major components: geometric modeling and motion control; cognitive modeling that enables Virtual Users to understand the environment model; and behavioral modeling that seamlessly integrates sources of theoretical and practical environment-behavior studies, statistics from a field study, and an Artificial Life approach. By inserting the Virtual Users into our environment model and letting them "explore" it on their own volition, our system reveals the interrelationship between the environment and its users.

1. Introduction

Environmental behavior simulation can be used to predict and evaluate the impact of environments on human behavior and it is of great interest to designers and clients. User modeling (as well as environment modeling) is key to such simulations. We have developed a user model, which we call a Virtual User, defining the goals, social traits, perception, and physical behaviors of each user in a simulated environment. Virtual Users are modeled as autonomous agents, which have the ability to 'understand' their environment and behave accordingly. Compared with existing user simulations, our model represents a new approach of integrating an agent-based model and a statistical behavior model into a coherent user model, Section 2.

J.S. Gero (ed.), Design Computing and Cognition '06, 61–79.
© 2006 *Springer. Printed in the Netherlands.*

The Virtual User model includes three major components: (1) geometric modeling and motion control, (2) cognitive modeling, and (3) behavioral modeling, Figure 1.

Geometry modeling represents Virtual Users as mannequins, with articulated body geometry, texture mapping and animation. To achieve autonomy, a good strategy for our purpose is to encapsulate basic motions (walking, running, and sitting, etc.) within the user models and enable script control of series of motions. Virtual Users' autonomous movements can then be controlled through high-level behavior rules (detailed in Section 3).

Cognitive modeling defines the Virtual Users' accessibility to the environment model. Perceiving and understanding environments are the prerequisite for Virtual Users to behave properly. However, simply providing all the information of the environment models to each user will not solve the dynamic problems that are not predictable in advance, such as encountering another moving user so that they can avoid collision. Our solution to perception is the combination of four components: "seeing" their local environment, "knowing" the global environment, "finding" paths to destinations, and "counting" duration of a specific behavior (detailed in Section 4).

Behavior modeling is the most critical issue underlying the simulation because it must mimic closely how humans behave in similar socio/spatial environments, given similar goals. Accordingly, our behavior modeling stems from three important and firm sources in different research areas: theoretical and practical environment-behavior studies, real world data from a field study, and an Artificial Life approach. A seamless integration of these sources into a working solution results in behavior simulation that is close to reality (detailed in Section 5).

We have conducted a case study with a campus plaza – Sproul Plaza at the University of California at Berkeley, which contains distinctive paving, a fountain with low seating edge, large area of steps, and a few benches, Figure 2. We first used video tracking in the plaza and obtained a large number of statistical data about people's behavior (Yan and Forsyth 2005) and then integrated the statistical data into our user modeling. By inserting the Virtual Users in our usability-based building model and letting them "explore" it on their own volition, our system reveals the interrelationship between the environment and its users (detailed in Section 6).

2. Existing User Models and Our Approach

Human spatial behavior simulations that exist are often limited to some well-defined areas of human activities where there has been considerable empirical research that can help develop the requisite cognitive models

(Kalay 2004). Some of the areas for which such cognitive models have been developed are pedestrian simulation and fire egress simulation.

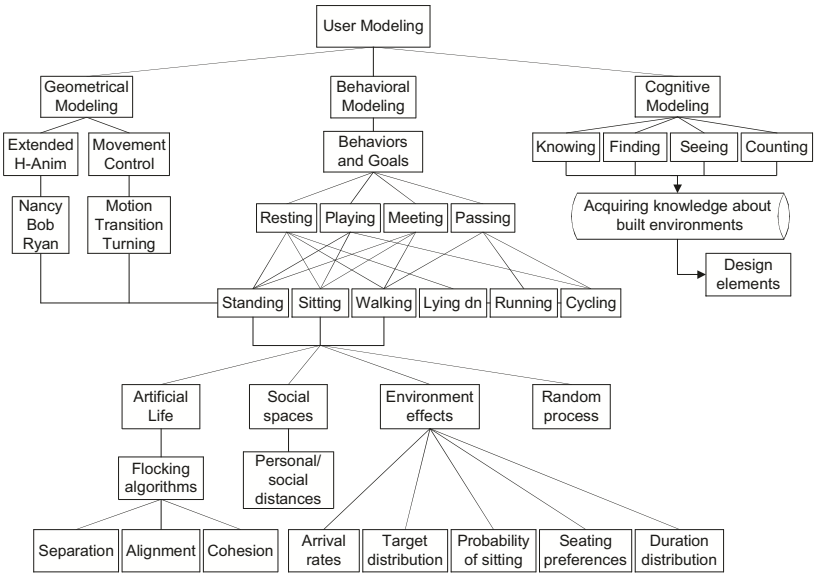


Figure 1. User modeling components: geometric modeling and motion control, cognitive modeling, and behavioral modeling.

Figure 2. Sproul Plaza at the University of California at Berkeley.

These simulations are often aimed at testing the Level of Service (Fruin 1971)—the amount of space people need to conduct certain activities, such as walking through corridors and doors, under normal or emergency situations. General human spatial behavior simulation models have been developed by Archea (1977), Glaser and Cavallin-Calanche (1999), and Kaplan and Kaplan (1982). They typically use discrete event simulation methods, where a generalized algorithm tracks minute-by-minute changes, geometry-based approaches (Glaser and Cavallin-Calanche 1999), or neural-nets (O'Neill 1992).

Batty (2001) has given a good review of recent pedestrian behavior modeling that employs agent-based models. These models take a very different view of probability than that used in more traditional transport and traffic models. Most transportation projects model movement patterns at a much higher scale than that of walking and they are not applicable at the kinds of fine scale that are associated with pedestrian movement. The advantages and feasibilities of using agent-based behavior simulation model include the following (Batty 2001): computer programming has become more object-oriented with individual events and artifacts being treated as classes whose behavior can be explicitly simulated; and new ways of articulating social systems by using ideas from complexity theory have developed over the past years, e.g. by Gilbert and Doran (1994). Some agent models have plans giving distinct purpose to their trips that drive them to complete some tasks, such as shopping (Haklay et al. 2001; Kerridge et al. 2001). Some other models are derived from various analogies in fluid dynamics and particle systems and also embracing key ideas from the theory of self-organization. All models emphasize the way pedestrians interact with one another and with the environment they walk in (e.g. Helbing et al. 1997). The general rules these models use are walking rules for interpersonal and obstacle avoidance and shortest path (e.g. Helbing et al. 2001).

Stahl (1982) and Ozel (1993) developed fire egress models, simulating the behavior in emergency. Ozel's model uses actions (such as "go to exit") and goal modifiers (such as "alarm sounds") libraries to define the behavior rules. These libraries, in turn, use the fire event, the building configuration, and the characteristics of the people as the determinants of their rules.

For various purposes, such as industrial product design, entertainment, and medicine, researchers have created many human models. Developed at the University of Pennsylvania beginning in 1984, Jack is a human model used in industry and government, showing car manufacturers whether their designs can accommodate a person of a certain size, or construction companies whether a particular task might leave employees injured and unproductive. Jack is good at testing ergonomics of a product design, while our Virtual Users are specifically created for evaluating environments. Thalman et al. have built virtual humans and an Informed Environment that

creates a database dedicated to urban life simulation. Using a set of manipulation tools, the database permits integration of what they call "urban knowledge" in order to simulate more realistic behavior. Moreover, for various types of mobile entities, they can compute paths to move through the city according to area rules. By using data derived from the environment, virtual humans are able to acquire urban behavior (Thalmann et al. 1999). Therakomen (2001)'s simulation uses agent-based model (Artificial Life) that is also used in our user model.

However, these simulations lack components that we think are essential to environmental behavior simulation: the integration of an agent-based model, which employs human social/personal space rules, and a statistical behavior model, which provides goal distribution and overall behavior patterns, into a coherent user model (as well as a systematic approach for creating usability-based environment model, which is described in Yan and Kalay 2005).

Based on Steinfeld (1992) and Kalay and Irazábal (1995)'s proposals toward an artificial or Virtual User, we first proposed and then developed a new computational model that simulates a built environment, its occupants and their behavior. We have developed methodologies and algorithms to build the simulation that consists of a usability-based building model and an agent-based user model. The building model is a discrete spatial model that represents the building objects. It possesses both geometric information of design elements and non-geometric information about the usability properties of these elements. The relationship between design elements and their intended users, which was implicitly understood by the designer, now becomes explicit to the Virtual Users through this usability-based modeling.

The agent-based user model is a computer model that defines behavioral rules for each individual to simulate both individual and group behavioral patterns including encountering, congregating, avoiding, interacting, etc. The behavior rules are derived from previous literature of human spatial behavior, a field study, and Artificial Life research. The agent-based user models are autonomous models. They emulate the realistic appearance, movement, and perception of individual users under normal conditions. They have adjustable profiles that consist of physical and social variables. The simulation of group behavior is pursued simultaneously with the simulation of individual behavior, and is achieved automatically by aggregating individual behavior without extra efforts. The methods of this simulation are described in the following sections.

3. Geometric Modeling of Virtual Users

Geometric modeling includes 2D and 3D modeling. 2D modeling is used for behavioral simulation and 3D modeling is used for behavioral visualisation.

3.1. 2D MODELING

The Virtual User's 2D model is a fairly abstract symbol used for user model design and checking purposes in the simulation phase. The Virtual User (as well as our environment model) utilises Scalable Vector Graphics (SVG) format – a graphical presentation of XML – for the purpose of presenting both geometrical and non-geometrical information. As shown in Figure 3, its graphical view is made of a filled circle and a short line indicating the facing direction of a Virtual User.

Figure 3. Virtual User's 2D model.

Its textual view represents non-geometrical information of Virtual Users' traits, as shown in Table 1.

TABLE 1. Content of the Virtual User's model in SVG format.

```
<svg>
        <title>Plaza User</title>
        <g id='geom' style='stroke:blue;stroke-width:0.1;fill:#00ff00;'
transform='translate(0,0) rotate(0,0,0)'>
        <circle r='.375'/>
        <line x1='0' y1='0' x2='0.75' y2='0' style='stroke:red;stroke-width:0.1;'>
        </g>
        <traits
                id='vuser'
                …….

                prob_1_5='1.58'
                …….
                time_sit_fountain_mean='190'
                time_sit_fountain_std='124'
                …….
                >
        </traits>
</svg>
```

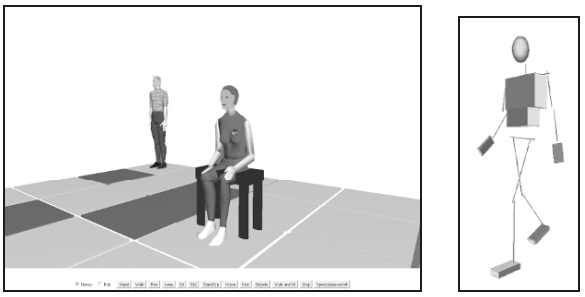
In the above table, a Virtual User's geometry is defined by a circle and a line with transformation data. The user's traits are defined based on our field study – a large number of statistical data obtained by video tracking (Yan and Forsyth 2005), including probabilities of users' choices of sitting by the fountain, on the steps, or on the benches, respectively; their duration of stay; their walking paths, etc. For example, "prob_1_5" in the table means the probability of a user comes from Lower Plaza and chooses to go to the

fountain, and its value is 1.58 %; and "time_sit_fountain_mean" is the average time that users spent sitting by the fountain and its value is 190 second. The standard deviation of sitting duration by the fountain is 124 second.

3.2. 3D MODELING AND MOTION CONTROL

The Virtual User's 3D model (as well as the environment's 3D model) utilises VRML for seamless integration of the two models in visualisation.

The user's 3D VRML model represents Virtual Users as mannequins, with articulated body geometry, texture mapping and animation, and conforms to the international standard of human modeling—Humanoid Animation Specification (H-Anim, 1.1, by Human Animation Working Group). It is used to represent realistic close-up models of Virtual Users, their walking and sitting animations, Figure 4(a), (Ballreich 1997; Babski 1998; and Lewis 1997).



(a) (b)

Figure 4. (a) Nancy and Bob demonstrating sitting and standing behaviors respectively, (b) Ryan demonstrating sitting behavior.

This model, however, is computationally too expensive for visualizing groups of people. Therefore, we created a simpler human model called Ryan, based on low-level limb movements that are encapsulated within the H-Anim model, such as the arms' and legs' movements for walking. These stick-figures have the same high-level movements as the close-up models (walking, running, and sitting), without the overhead of fully fleshed-out bodies, Figure 4(b).

Our simulation has shown that the Ryan model has the minimal level of details needed to depict behavior patterns: we can see clearly how people use the public space using these models. The Nancy and Bob models with higher level of details require considerable computational resources. By using them, we can achieve a more realistic visualisation that is similar or even better than that in the video used in our field study, which has been shown in Figure 2.

The H-Anim models use prototype design concept (PROTO in VRML). In H-Anim models, human's joints (e.g. a shoulder) and segments (e.g. an upper arm) are defined as PROTOs with field types, data types, field names, and default values (see Ames et al. 1997 for details of PROTO in VRML). Virtual User model extends the Humanoid PROTO in H-Anim model by adding clothes colors for distinguishing individual Virtual Users and motion control variables for triggering behaviors such as standing, walking, running, standing up, and sitting down.

By augmenting the VRML model with Java programming, we created a real-time motion control to start or stop walking, running, sitting down, sitting still, standing up, and standing still. The control makes it possible to create a sequence of motions using a script, e.g. walk to location X, sit for n minutes, and walk to location Y. Transitional movements such as sitting down and standing up are inserted into motion sequence automatically. For example, if a Virtual User first walks and then sits, a transition of sitting down is inserted between walking and sitting.

Each Virtual User model occupies a cell in the discrete space model and has 8 directions: north, south, west, east, northeast, northwest, southeast, and southwest. Turning is calculated automatically so that there is no need to specify it: whenever a Virtual User starts a journey in a new direction, it will turn along the direction smoothly and go forward, just like a real user.

4. Cognitive Modeling of Virtual Users

Cognitive modeling defines the users' ability to access and interpret the environment model. Perceiving and understanding environments are the prerequisite for Virtual Users to behave properly. However, simply providing all the information of the environment models to each user will not solve the dynamic problems that are not predictable in advance, such as encountering another moving user so that they can avoid collision. Our solution to perception is the combination of four components: "seeing" their local environment, "knowing" the global environment, "finding" paths to destinations, and "counting" the duration of a specific behavior.

4.1. KNOWING

"Knowing" the entire environment in advance to help make basic decision of what to do and how to behave, much like a frequent visitor has good knowledge of the space. The Virtual Users enter the plaza with knowledge of all the design elements, e.g. the starting points and the targets, which are entrances or openings of the plaza. They know each cell's design element type, be it ground, steps, fountain-side, fountain water, benches, etc. They know the location and orientation of all seats, so they can seek them out in order to sit on one. They also know the cells that are obstacles they need to

avoid on their journeys such as the fountain and the benches. They need to calculate the shortest paths to go to their destinations using a search algorithm.

4.2. FINDING

"Finding" paths to destinations. People are naturally very conscious of their choice of routes because it is generally tiring to walk. If the target is in sight, they tend to steer directly toward it, sometimes crossing the plazas diagonally. Observations show that almost everyone follows the shortest routes across plazas; only users who push bicycles or baby carriages make detours (Gehl 1987). For the Virtual Users, we employed A* algorithm for searching the shortest path. A* algorithm is widely used in games (Russell and Norvig 1995). We optimized A* in our simulation to reduce the search space from the total number of cells to a subset of cells. Given starting point, target point, and original empty cells and obstacles, the algorithm first returns a subset of the empty cells and obstacles. The subset is defined as a rectangular area with starting and target points as corners. That way the search space is very much reduced and the performance of searching is speeded up significantly. In most cases a Virtual User can find a path in this subset of the search space. In case a shortest path can't be found in the subspace, the original search space will be used. See Section 5.5.1 for a sample of path finding using A* algorithm.

4.3. SEEING

"Seeing," i.e. accessing the relevant parts of the environment model within circular areas (social spaces) in front of a user in real-time, and translating them into terms that correspond to the Virtual User's cognitive model, for such purposes as avoiding collisions and recognizing an acquaintance or an object. Once a Virtual User obtained a path, it will start to walk on each cell along the path. During its journey, the Virtual User needs to avoid hitting others and to keep reasonable inter-personal distance. It will stick to the path unless someone else comes close. At each step the Virtual User checks its social spaces. If others are found within the spaces, the Virtual User will adjust its path. In case of meeting acquaintances, a user will stand still (and talk with them for a while). All of these kinds of knowledge are obtained in real time. (Social spaces are detailed in Section 5.2.)

4.4. COUNTING

"Counting" the duration of a specific behavior, such as sitting, to make a decision about what to do next: continue sitting if the duration has not exceeded a preset maximum duration based on our statistics (from the field study), or walk away.

5. Behavioral Modeling of Virtual Users

Behavioral modeling is the most critical issue underlying the simulation because it must mimic closely how humans behave in similar socio/spatial environments, given similar goals. Accordingly, our behavioral modeling is based on three important and firm sources. The first source includes theoretical and practical environment-behavior studies, such as those by Lewin (1936), Moore (1987), Stokols (1977), Hall (1966), Whyte (1980), Gehl (1987), etc. The common characteristics of these theories provided us with the basic relationship between environment and behavior. The relationship can be expressed as: B = f (G, R, E), where G, R, and E stand for the goals, behavior rules, and the built environment, respectively. Goals are high-level objectives, the results of intra-personal processes. Rules are the results of physiological and psychological processes, influenced by social and cultural factors. The built environment is comprised of design elements.

The second source of data is our field study using video tracking, which provided important and substantial statistical measurements about users' behavior, e.g. users' goals and overall behavior patterns (Yan and Forsyth 2005).

The third source of data is Artificial Life research, which provided primitive group behavior algorithms. Built from simple behavior rules for individual users, the group behavior algorithms are used for simulating spatial interactions among individuals during their movements.

Using these three sources, we developed an agent-based approach, where the behavior of Virtual Users (which include walking through the plaza, sitting by the fountain, on the benches, or on the steps, or standing while meeting acquaintances, etc.), is determined through a hierarchical structure of rules, Figure 5, which resulted directly from the following aspects.

5.1. ARTIFICIAL LIFE APPROACH

The Virtual Users' primary movement control is inspired by Artificial Life's flocking algorithm (Reynolds 1987). Three simple rules define the heading direction of a so-called Boid and result in a complex behavior pattern that mimics birds' flocking. The three rules are:

(a) **Separation** - steering to avoid crowding local flockmates, Figure 6, left;
(b) **Alignment** - steering towards the average heading of local flockmates, Figure 6, middle; and
(c) **Cohesion** - steering to move toward the average position of local flockmates, Figure 6, right.

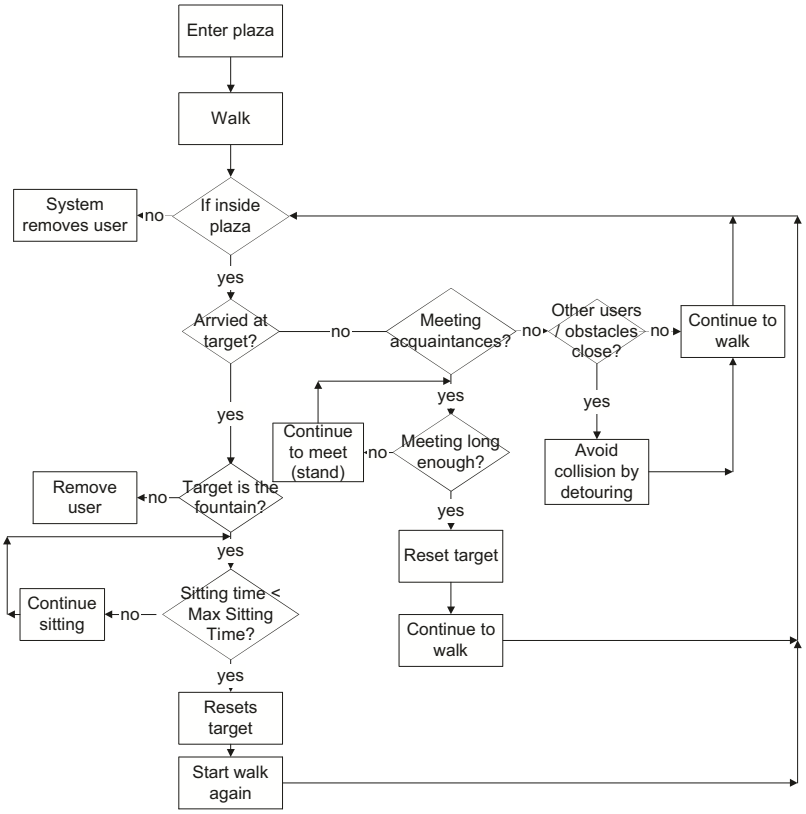


Figure 5. Hierarchical structure of Virtual Users' behavior rules.

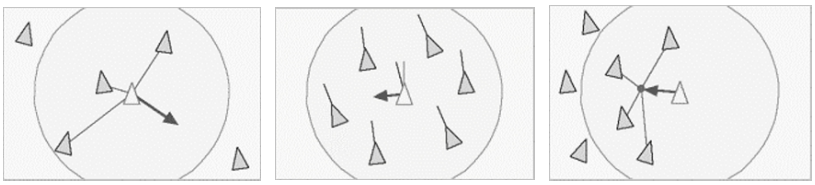


Figure 6. Boids' flocking algorithm. Reynolds (1999).

5.2. SOCIAL SPACES

Environment-behavior studies validated and helped apply Artificial Life's flocking algorithm to users' behavior simulation in public spaces. When applied to user simulation, Artificial Life's flocking algorithm is modified with consideration of human social environmental factors.

5.2.1. Separation

On plazas, the closeness is gratuitous (Whyte 1980), which means people want to keep certain distances from each other. They try to avoid collision with other people of the same or different directions on their paths. They also defer to someone of higher priority if in conflict with priority determined by age or gender (Gehl 1987). Different kinds of distances among people, discovered by Hall (1966), are used to determine the minimal distance between users:

(a) **Intimate Distance**: (6 ~ 18 inches)
(b) **Personal Distance**: close phase (1.5 ~ 2.5 feet), far phase (2.5 ~ 4 feet)
(c) **Social Distances**: close phase (4 ~ 7 feet), far phase (7 ~12 feet)
(d) **Public Distances**: close phase (20 ~ 25 feet), far phase (25 or more)

For a graphical illustration about the distances, Figure 7.

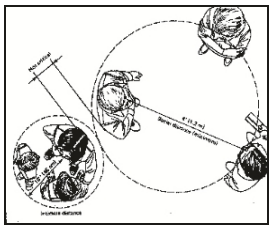


Figure 7. Personal-space bubbles. Source: Deasy (1985).

5.2.2. Alignment

Whyte (1980) observed the following pedestrian behavior in public spaces: people walking quickly, walking slowly, skipping up steps, weaving in and out on crossing patterns, accelerating and retarding to match the moves of the others.

Gehl (1987) also found that pedestrians align in two-way traffic. The upper limit for two-way pedestrian traffic is 10 –15 pedestrians per minute per meter street width. If the number increases, the tendency of dividing into two parallel opposite streams occurs. People start to keep to right, and freedom of movement is more or less lost. In a bi-directional pathway, passing on the right-hand side (which forces alignment) is a rule in countries such as US, etc. and left-hand side in UK, etc.

5.2.3. Cohesion

What attract people most are other people and their activities. People try to stay in the main pedestrian flow or move into it (Whyte 1980). They gather

with and move about with others and seek to place themselves near others. New activities begin in the vicinity of events that are already in progress (Gehl 1987). This is the so-called self-congestion behavior.

From the above comparison we believe that it is reasonable to apply the Artificial Life's flocking algorithms to simulating users' movement in a plaza, with consideration of human's social and spatial factors, and the environmental effects (which will be discussed in the next section). Using Hall's proxemics findings, we created social spaces for a Virtual User by grouping the cells in front of a user into different spaces, Figure 8. As a Virtual User changes its direction, the spaces change as well. The corresponding parameters used are shown in Table 2. The spaces used that affect Virtual Users' movement are personal space, social space's close phase, and social space's far phase. Public distance is not affecting users' movement because other persons' present can be seen only peripherally in this distance (Hall 1966).

When users move in the plaza, at each step they will check whether there are other users or design elements are invading their social/personal spaces and if so, they will behave accordingly, e.g. to stop and stand to talk if meeting acquaintances, or detour if meeting strangers or obstacles.

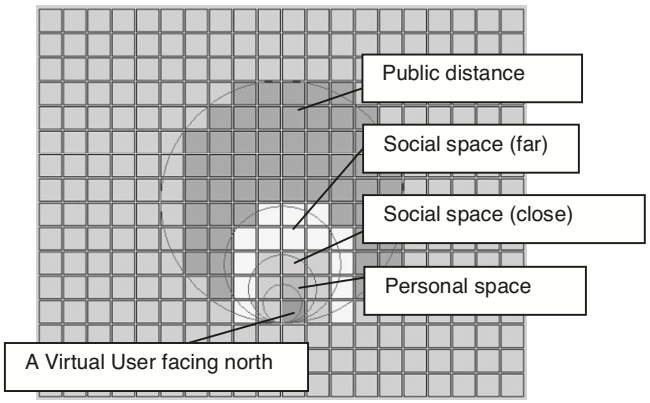


Figure 8. Virtual users' social spaces. Each cell in the grid of the usability-based building model is an object that possesses several layers of usability properties.

TABLE 2. Parameters of users' social spaces (Java implementation).

```
//distances of personal/social spaces
size = tile.size / scale; //750 mm
int personalDistance = 1200; //mm, 4 feet
int socialDistanceCloser = 2100; //mm, closer social distance, 7 feet
int socialDistanceFarther = 3600; //mm, farther social distance, 12 feet
int publicDistance = 7500; //mm, 25 feet;
```

5.3. ENVIRONMENTAL EFFECTS

Our field study provided the model with users' goals and quantitative measurements of overall behavior patterns, including:

(a) **Entering rates** to set up the frequency of inserting Virtual Users into the plaza from different entrances, based on (1) total number of people who entered our observation region in the plaza during the time interval of our field study; and (2) Poisson distribution. Poisson distribution is commonly used to model the number of random occurrences of some phenomenon in a specified unit of space or time. (For more details of Poisson distribution, see Spiegel 1992). Thus it is a good choice to use Poisson distribution to model the users' entering rate. We have also found that the distribution of the arrival rate per minute during the time interval is close to a Poisson distribution.

(b) **Target distribution** based on numbers of users walking in different routes and their probabilities. We applied the probabilities of a user entering from one entry and exiting from another or heading to a seat.

(c) **Probabilities of users choosing to sit** based on numbers of people who entered the plaza chose to sit vs. to walk crossing the plaza.

(d) **Seating preferences** based on people's choices among fountain, benches, and steps.

(e) **Distribution of duration** with means and standard deviations of duration at different seating places.

5.4. RANDOMIZATION

To add more realism to behavior simulation, we applied random processes to model users' behavior patterns.

(a) **Poisson distribution** is used to set up the rate that users enter the plaza. It is also confirmed by our field study.

(b) **Normal distribution** is used for duration of sitting and standing.

(c) **Uniform distribution** is used in the following processes:

- Random appearance (clothes colors) to differentiate Virtual Users in visualisation.
- Random starting or ending points at entering or exiting areas.
- Probability of meeting acquaintances (a Virtual User will stand still then).

5.5. SAMPLE SCENARIOS

Combining all the four components: Artificial Life algorithms, social spaces, environmental effects, and randomization, we built a user model to simulate

individual and group behaviors. The implementation details will be discussed in Section 6.

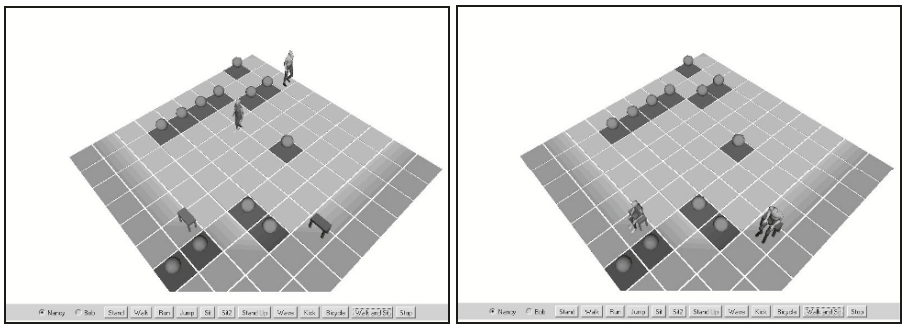
The following two scenarios are intended to test how the behavior simulation works. They reveal that many behavior patterns can be simulated. For testing purposes, we used only two Virtual Users: Bob and Nancy.

5.5.1. Finding benches in a plaza

Nancy and Bob use the shortest path search algorithm (A*) to find benches to sit on. Greenery is treated as an obstacle they must circumnavigate. Nancy has priority over Bob to get her nearest bench. So when a given bench is identified as the nearest seat from both Nancy's and Bob's points of view, Nancy will get it and Bob will look for another seat, even if it is further than the first one. The graphical user interface allows designers to move the benches and let Nancy and Bob find them, Figure 9.

5.5.2. To sit in the sun or in the shade?

For each cell of the space we calculated dynamically whether it is in the sun or in the shade, based on the plaza's geographic location, the sun's azimuth and altitude at a given time, and objects such as trees and buildings that may cast a shadow on the ground. Virtual Users can 'know' a tile's sun/shade disposition, and choose whether to sit on a bench located on that cell or not. Figure 10 shows that Nancy prefers a seat in the sun rather than a seat in the shade, even though the one in the shade may have been closer to her point of departure.



(a) (b)

Figure 9. (a) Nancy and Bob started to walk to the benches using A* search algorithm, (b) Nancy and Bob found the benches and sat.

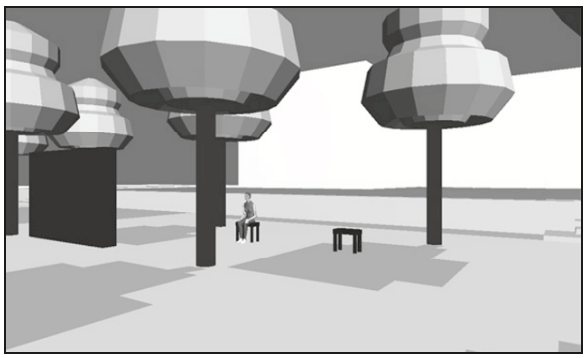


Figure 10. Nancy prefers a seat in the sun to a seat in the shade.

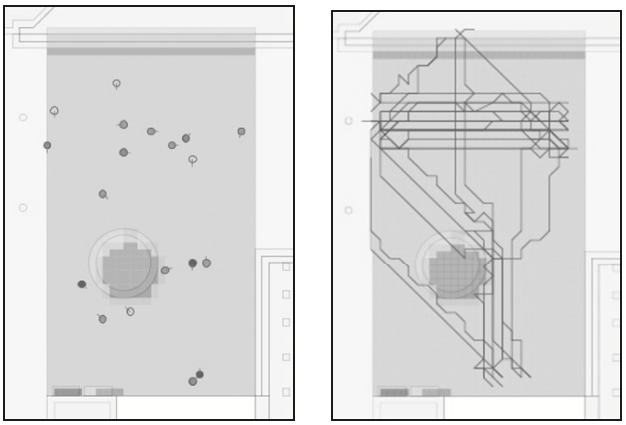
6. Applying the User Model to Behavior Simulation

We applied the user model to our behavior simulation in the run time through a simulation engine.

The simulation engine first loads the building model and parses the model's graphical and usability properties, then creates a Virtual User group—a list that allows an unlimited number of user models to be added into, and upon completion of a journey removed from the list. The engine runs the simulation step by step, and at each time step (one second) it adds users from the entrances and moves all the users by one step. The Virtual Users acquire environmental knowledge through the cognitive processes (knowing, seeing, finding, and counting) so that the users know, for example, where they can walk and where they can sit. Then the engine lets the Virtual Users move following behavior rules, e.g. shortest path, group movement rules, and social spaces. The simulation engine uses Batik SVG toolkit with Java2D rendering engine, and Document Object Model (DOM) to traverse the design element tree of the building model (see Watt et al. 2003 for details of Batik SVG toolkit).

The simulation results include (1) a 2D animation of Virtual Users movements, including walking and standing in the plaza, sitting at different places, and meeting other users, etc. Figure 11(a), and (2) a behavior data set that records all users' behavior information associated with their paths, including the coordinates along paths, arrival time, motions (walking, sitting, or standing), sitting directions, and duration of stay, Figure 11(b).

.



(a) (b)

Figure 11. (a) 2D animation of Virtual Users' movements. (b) Virtual Users' paths drawn using dark lines.

Finally, by inserting 3D models of Virtual User into a 3D model of the plaza and letting the users move following the behavior data recorded in the simulation, we realized behavior visualisation—animations in which Virtual Users exhibit similar traits to those observed in reality: walking, sitting, meeting other Virtual Users, etc., Figure 12.

7. Conclusion

Our model supports fast creation of realistic user simulation because the Virtual Users are re-usable, autonomous constructs, and their behaviors are driven by adjustable variables of users' characteristics and spatial configurations.

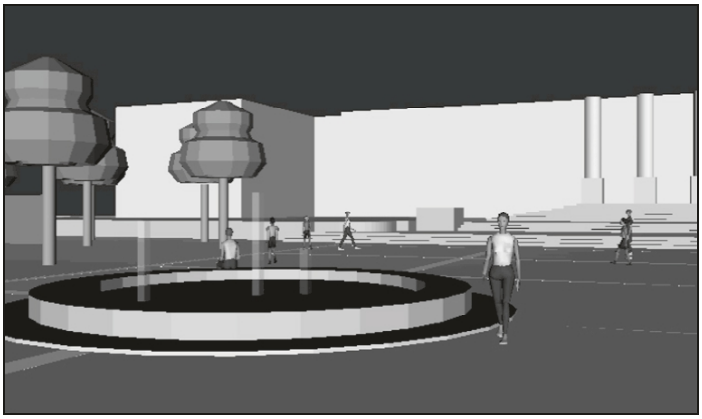


Figure 12. 3D visualisation: Virtual Users exhibit similar traits to those observed in reality (walking, sitting, meeting other Virtual Users, etc.).

We expect, with our Virtual User simulation, human behavior analysis as one of the most important aspects in building design can be integrated into designers' daily design practices seamlessly. The evaluation of human spatial behavior can be made easier and visible before the building is built. This will encourage designers to pay more attention to users and therefore innovative buildings concerning more about the needs of people can be designed and built.

References

Ames, A, Nadeau, DR, Moreland, JL:1997, *VRML 2.0 Sourcebook,* Second Edition. John Wiley & Sons, Inc.

Archea, J: 1977, The place of architectural factors in behavioral theories of privacy. *Journal of Social Issues*, **33**(3): 116-137.

Ballreich, C: 1997, nancy.wrl, open source program, copyright 3Name3D/Yglesias, Wallock, Divekar, Inc. Available Online: http://www.ballreich.net/vrml/h-anim/h-anim-examples.html

Babski, C: 1998, Baxter.wrl, open source program, copyright LIG/EPFL. Available Online: http://ligwww.epfl.ch/~babski/StandardBody/

Batty, M: 2001, Editorial: Agent-based pedestrian modeling, *Environment and Planning B: Planning and Design* **28**(3): 321-326.

Deasy, CM and Lasswell, TE: 1985, *Designing Places for People: A Handbook on Human Behavior for Architects, Designers, and Facility Managers*, Whitney Library of Design, New York.

Fruin, JJ: 1971, *Pedestrian Planning and Design*, Metropolitan Association of Urban Designers and Environmental Planners, New York.

Gehl, J: 1987, *Life Between Buildings: Using Public Space*, English translation, Van Nostrand Reinhold, New York.

Gilbert, N and Doran, J (eds): 1994, *Simulating Societies*, UCL Press, London.

Glaser, DC and Cavallin-Calanche, HE: 1999, Representations and meanings: Cognitive models for simulations of architectural space, *in* W Porter and G Goldschmidt (eds), *4th International Design Thinking Research Symposium on Design Representation*, MIT, Cambridge, MA, pp. 129-144.

Haklay, M, O'Sullivan, D: 2001, So go downtown: Simulating pedestrian movement in town centres, *Environment and Planning B: Planning and Design* **28**(3): 343-359.

Hall, ET: 1966, *Hidden Dimension*, Doubleday, New York.

Helbing, D, Schweitzer, F: 1997, Active walker model for the formation of human and animal trail system, *Physical Review E* **56**: 2527-2539.

Helbing, D, Molnar, P: 2001, Self-organizing pedestrian movement, *Environment and Planning B: Planning and Design* **28**(3): 361-383.

Kalay, YE: 2004, *Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design*, MIT Press, Cambridge, MA.

Kalay, YE and Irazábal, CE: 1995, Virtual Users (VUsers): Auto-animated human-forms for representation and evaluation of behaviour, *in* Designed Environments, *Technical Report*, University of California, Berkeley.

Kaplan, S and Kaplan, R: 1982, *Cognition and Environment: Functioning in an Uncertain World*, Prager, New York.

Kerridge, J, Hine, J: 2001, Agent-based modeling of pedestrian movements: The questions that need to be asked, *Environment and Planning B: Planning and Design* **28**(3): 327-341.

Lewin, K: 1936, *Principles of Topological Psychology*, New York, London, McGraw-Hill book company inc.

Lewis, M: 1997, nancySit.wrl, open source program, ccopyright 3Name3D/Yglesias, Wallock, Divekar, Inc. Available Online: http://www.accad.ohio-state.edu/~mlewis/VRML/Sanbaso/Nancy/

Moore, GT: 1987, Environment and behavior research *in* North America: History, developments, and unresolved issues, *in* D Stokols and I Altman (eds), *Handbook of Environmental Psychology*, John Wiley, New York, pp. 1371-1410.

O'Neill, MJ: 1992, A neural network simulation as a computer-aided design tool for evaluating building legibility, *in* YE Kalay (ed), *Evaluating and Predicting Design Performance*, John Wiley, New York. pp. 347-366.

Ozel, F: 1993, Computer simulation of behavior in spaces, *in* D Stokols and RW Marans (eds), *Environmental Simulation - Research and Policy Issues*, Plenum Press, New York, pp. 191-212.

Reynolds, CW: 1987, Flocks, herds, and schools: A distributed behavioral model, *Computer Graphics* **21**(4): 25-34.

Reynolds, CW: 1999, Steering behaviors for autonomous characters, *Proceedings of Game Developers Conference,* San Jose, California. Miller Freeman Game Group, San Francisco, California, pp. 763-782.

Russell, S and Norvig, P: 1995, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey.

Spiegel, MR: 1992, *Theory and Problems of Probability and Statistics*, McGraw-Hill, New York.

Stahl, F: 1982, Computer simulation modeling for informed design decision making, *in* P Bart, A Chen and G Francescat (eds), *13th Annual Conference of the Environmental Design Research Association*, Washington, DC. EDRA, pp. 105-111.

Steinfeld, E: 1992, Toward artificial users, *in* Y E Kalay (ed), *Evaluating and predicting design performance*, John Wiley, New York, pp. 329-346.

Stokols, D: 1977, Origins and directions of environment-behavioral research, *in* D Stokols (ed), *Perspectives on Environment and Behavior*, Plenum Press, New York, pp. 5-36.

Thalmann, D, Farenc, N: 1999, Virtual human life simulation and database: Why and how, *in* Y Kambayashi and H Takakura (eds), *International Symposium on Database Applications in Non-Traditional Environments*, IEEE CS Press, Japan, pp. 63-71.

Therakomen P: 2001, *Mouse.class: Experiments for Exploring Dynamic Behaviors in Urban Places*. M.Arch Thesis, Department of Architecture, University of Washington.

Watt, A: 2003, *SVG Unleashed*, Sams Publishing.

Whyte, W: 1980, *The Social Life of Small Urban Spaces*, Project for Public Spaces, New York.

Yan, W and Forsyth, D: 2005, Learning the behavior of users in a public space through video tracking, *IEEE Workshop on Applications of Computer Vision*, Breckenridge Colorado, pp. 370-377.

Yan, W and Kalay, YE: 2005, Simulating human behavior in built environments, *CAAD Futures*, Vienna, Austria, pp. 301-310.

EARLY STAGES OF DESIGN

Exploration through drawings in the conceptual stage of product design
Miquel Prats and Chris F Earl

Digital sketching in a multi-actor environment
Alexander Koutamanis

The Designosaur and the furniture factory
Yeonjoo Oh, Gabriel Johnson, Mark Gross and Ellen Yi-Luen Do

Industrial mechanical design: the “ids” case study
Stefania Bandini and Fabio Sartori

EXPLORATION THROUGH DRAWINGS IN THE CONCEPTUAL STAGE OF PRODUCT DESIGN

MIQUEL PRATS AND CHRIS F EARL
The Open University, UK

Abstract. This paper argues that sequences of exploratory drawings - constructed by designer's movements and decisions - trace systematic and logical paths from ideas to designs. This argument has three parts. First, sequences of exploratory sketches produced by product designers, against the same task specification, are analyzed in terms of the cognitive categories of reinterpretation, emergence and abstraction. Second, a computational model is outlined for the process of exploration through drawing and third the model is applied to elucidate the logic in the sequences of exploratory sketches examined earlier.

1. Introduction

Designers rely on visual representations to generate and explore design ideas. It is assumed here that there is a reciprocal relationship between designers' thinking and their representations. Representations may be a consequence of thinking but also thinking may be stimulated by perception of representations. These connections suggest that explorations might trace systematic and logical paths from an original idea to a final design via sequences of drawings and decisions.

Visual representations, particularly freehand sketches, serve as a tool to assist thinking (Goel 1995). Sketches in conceptual stages support both divergent and convergent thinking. While in divergent thinking designers generate isolated concepts, in convergent thinking these are synthesized and evaluated (Liu and Bligh 2003). Cross (1994) points out that, in general, the design process is convergent, but also contains deliberate divergent thinking for the purpose of opening the search for new concepts. This paper concentrates on sketches generated during convergent thinking.

Generally sketches produced in early stages tend to be more ambiguous and less complex than later stages of the design process (Lawson 2004). Goel (1995) argues that, on the one hand, designers need ambiguous and vague sketches in order to keep their options open. But, on the other hand,

J.S. Gero (ed.), Design Computing and Cognition '06, 83–102.
© 2006 *Springer. Printed in the Netherlands.*

designers direct the design to a conclusion by setting boundaries, selecting particular objects and relations for attention, and imposing a coherence that guides subsequent moves (Schön 1988).

Designing includes reflective conversation with sketches in which designers proceed by seeing, moving and seeing again (Schön and Wiggins 1992). That is, interpreting, transforming and reinterpreting sketches. Goldschmidt (1994) notes that designers transform designs in a cyclic manner. Each sketch is transformed by adding, deleting, modifying or replacing parts. The reflective 'conversation' leads to the generation of sequences of related sketches.

The path leading to the final design cannot be foreseen, and each transitional design generated is a potential turning point where the path can change its course. How designers perceive shapes in drawings offers a point of departure in understanding the exploration process in design. How design features are perceptually grouped and transformed appears to be critical in understanding the exploration process of designs. Concentrating on shape as the medium for exploration, interpretations and transformations can be represented by shape rules (Stiny and Gips 1972) in a grammar which provides a connection between cognitive processes and formal exploration of designs. Figure 1 illustrates three shape rules used to explore the design shown in Figure 1(a) through using different interretations and transformations.

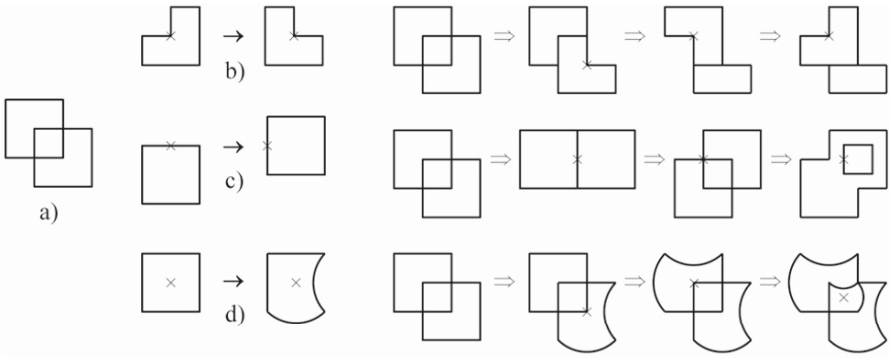


Figure 1. Shape rules encode different interpretations and transformations.

The left side of rules is used to decompose the design according to interpretation. The rule in Figure 1(b), for example, interprets the design as composed of L-shapes, and the rules in Figure 1(c) and Figure 1(d) interpret the design as composed of squares. The right side of shape rules is used to encode transformations of interpreted shapes. Some transformations involve changes in the structure of the design, Figure 1(b) and Figure 1(c), whilst others transform only the outlines keeping the structure constant Figure 1(d).

Shape grammar implementations have tried to set down generative specifications for styles (McCormack and Cagan 2004) or coherent sets of designs, however the free flowing exploratory capabilities of shape rules are rarely developed. Shape rules (Stiny 1980; Stiny 2006) have wider potential to bridge the gap between traditional sketching techniques and modern computational methods of design.

This paper has three parts. First, sequences of exploratory sketches produced by product designers, against the same task specification, are analysed in terms of the cognitive categories of reinterpretation, emergence and abstraction. Second, a computational model is outlined for the process of exploration through drawing and third the model is applied to elucidate the logic in the sequences of exploratory sketches examined earlier.

2. Visual Perception in Design Exploration

Perception is crucial in designing, and sketches provide designers with a basis to stimulate perception. The degree of a designer's ability to perceive their own sketches is a significant characteristic that differentiates experts from novices (Suwa 2005). In design exploration designers often use both imagery and visual perception simultaneously to explore new design alternatives (Goldschmidt 1994). Although both mechanisms are similar (Kosslyn 1990) their consequences may be different. While imagery allows exploring designs through the mind's eye, visual perception requires the support of visual stimulus such as sketches. Purcell and Gero (1998) review protocol studies of the roles of imagery and sketching in design. This paper concentrates on the exploratory sketches themselves and their perceptual consequences in design exploration stages.

Design sketches produced in exploratory stages are not always external representations of internal mental images. Rather they may be used as a way of thinking like talking out loud can be a way of thinking (Smithers 2001). The perception of sketches in design exploration assists designers in (i) inspecting compositions of designs as well as examining subtle features and (ii) discovering new design interpretations (Suwa and Tversky 2003). The review of these two processes offers a starting point for this paper.

Psychology and cognate disciplines aim to detect and understand general rules of perception. Arnheim (1974), for example, argues that many people see Figure 2(a) as unbalanced. Its composition looks accidental, transitory, and somewhat illogical. He points out that the circle is not only influenced by the boundaries of the square, but also by imaginary cross and diagonals that divide the square into symmetrical parts, which he refers to as the *structural skeleton*. The composition is more stable and settled when the circle and the square share the same centre, Figure 2(b). In general, when the

position of the circle coincides with a feature of the structural skeleton it appears balanced.

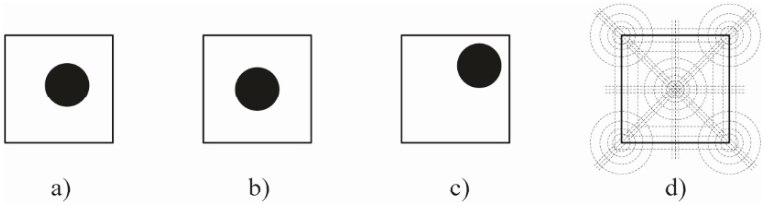


Figure 2. (a) Unbalanced composition, (b) and (c) more balanced compositions, (d) structural skeleton of the square.

Typically, sketches are composed of shape elements arranged relative to each other, and relative to a reference frame (Tversky 2001). The reference frame is similar to the idea of structure. Interpreting sketches involves grouping certain elements in a particular way and assigning a structure. Designers are sensible to this requirement when they arrange the elements in a design, and while exploring new designs they seek out the most suitable layouts of perceived elements. Akin (2001) argues that architects continue to search for alternative solutions even when they have developed satisfactory designs. One argument in favour of the existence of structure is that some shape transformations lead to refinements of the concept design whilst other types of transformations lead to different concept designs (Goel 1995). Therefore, some shape transformations may entail structure manipulation. Stacey (2005) points out the importance of structure in style judgments. Shared structure may appear more important than shared features, for example, the letter pair AA may be seen as more similar to BB than to AC.

The structure of a design is related to the perceptual organization of its elements. The structure assists in revealing interpretations of designs. Suwa and Tversky (2003) refer to constructive perception which involves organizing perception in the search for new interpretations. A structure can be used to guide the exploration of new designs. The arrangement of structures determines the identity of the pattern to such an extent that a given outline may produce completely different patterns depending on what structure is perceived in the design (Arnheim 1966). Visual perception is dynamic, and therefore, recognition of the structure of objects necessarily involves active participation by the viewer, as for example, proposed by Kepes (1944) for abstract paintings. Also, reversing figures, such as the duck-rabbit figure, offers a good example of this phenomenon.

Structures appear to be a crucial ingredient in understanding the mechanisms into design exploration. Observing design sketches can give an insight of the functioning of structures. However, this is difficult because structures are rarely explicitly represented in sketches. In order to overcome

this difficulty we examine three different cognitive mechanisms that, in some extent, are connected to structure. These are reinterpretation, emergence and abstraction.

Reinterpreting a shape entails changes in the structure. Generally, a shape can be interpreted differently if several structures fit the shape equally well (Arnheim 1966). Changes in interpretation often occur when relationships between elements are perceptually modified or emergent elements are discovered.

Discovering emergent elements in sketches seems to be vital in design exploration (Suwa and Gero 1999). The path design exploration takes will be altered whenever the structure is newly reinterpreted. Gero (1996) examines the role of emergence in creative design.

In the exploration of new design alternatives, designers modify the elements of the design according to a perceived structure. If structures are 'viewed' at a higher level of abstraction this reduces the complexity of designs and assists in understanding aesthetic properties, such as balance in composition. Designers switch between different levels of abstraction and use abstract models to test design decisions (Hoover and Rinderle 1991). They argue that while making a design refinement, the designer explicitly considers only those design characteristics which are included within the current abstraction. That is, shape refinements are made within the framework of the perceived structure.

In the next section, we examine sequences of sketches produced by designers during design exploration concentrating on the processes of reinterpretation, emergence and abstraction in the conceptual stage of product design and how structures are used in design exploration.

3. An Empirical Study of Design Exploration

Several studies have observed how professional designers and design students develop specific design activities. Goel (1995), for example, observed that, in convergent thinking, two types of strategies occur between successive sketches; lateral transformation and vertical transformations. While lateral transformations are used for widening the problem space by moving from one idea to a slightly different idea, vertical transformations deepen the design by moving from one idea to a more detailed or refined version of the same idea.

One way of examining designers' reasoning is by observation of their sketches through protocol analysis. However, many of these studies have focused on designers 'seeing' rather than designers 'moving' thus neglecting the investigation of shape relations among sketches. Here we examine the kinds of moving used in the early stages of product design. We conducted an

empirical study concerned with visual representations, particularly with what has been termed ‘thinking sketches’ (Ferguson 1992).

3.1. THE EXPERIMENT

There are different methods of investigating the design process and each has its strengths and weaknesses. For example, formal 'think-aloud' protocols, where participants are asked to review and talk through their work, is widely used for seeking insights into designers' creative activities such as sketching. In the presented experiment a more informal method is used. Participants developed a design task over four weeks, at their normal places of work, without being observed or forced to think-aloud. In order to accomplish these requirements the experiment was conducted by post after previous agreement with participants. They were provided with an introductory letter, an A3 sheet with an explanation of the task printed on the top left corner, and a questionnaire placed into another envelop, which participants had to open after the task. The results indicate patterns in the designer’s movements between sketches. These patterns have guided construction of a speculative model as the basis for further more formal experiment.

This informal approach was valuable because participants had the advantage of developing the task in their normal working places without the pressure of being videotaped. Participants had the possibility of breaking up the sketching process in various phases over the four weeks. Further, the fact that participants were provided with an ‘official’ sheet to sketch on induced some to sketch additional experimental and personal concepts on extra sheets. At the end of the task participants were asked to submit all sketches produced during the design process. Eight industrial designers were selected with broad professional experience including consumer products, packaging and urban furniture. All participants had proficient drawing skills.

Participants were asked to devise a design for a new electric jug kettle following a concise brief. They were encouraged to produce at least 10 sketches to come up with a single and preferred proposal. In order to analyse progression in designing, participants were asked to number the sketches as they created them and they were reminded not to erase anything. After completing the task, participants sent back the A3 sheet, and the questionnaire as well as all other representations developed during the design process.

3.2. OBSERVATIONS

Each designer produced on average 20 sketches. Most of these sketches are characterised by overtracing, in which the participant repeatedly outlined a particular shape or area of the sketch. According to Do and Gross (1996), overtracing serves several functions: (i) selection or drawing attention to an

element, (ii) shape emergence, attending to one or another shape interpretation, and (iii) shape refinement, or adding detail to an abstract or roughed out shape. The overtracing of sketches has assisted us in identifying where participants change interpretation and detect emergent shapes.

Most participants used brief annotations in their sketches indicating, for example, the position of buttons or the material of a specific part of the kettle. Text was also used by some participants in bubble diagrams or to name concepts, e.g. water drop, bamboo or gourd, identifying their own interpretations of the sketch. Although this experiment focuses on the shape of sketches, annotations of participants were particularly useful in the analysis of the reinterpretation of concept designs.

Participants varied the complexity of their sketches through the process. While some sketches had few lines and no details, others were produced with more detail including annotations, shades or hidden lines for example. The hypothesis here is that the level of complexity of sketches reflects the level of abstraction that designers perceive the concept design at that particular moment. We observed that abstract representations, as well as reinterpretation and emergence were salient characteristics that assisted designers in their exploration of designs.

3.2.1. Reinterpretation

Close observation of the sketches reveals features that suggest that participants performed changes of interpretations of their sketches. Changes of interpretation have been identified by comparing the type of strokes used among sketches that represent the same concept design. Van Sommers' (1984) experiments, for example, suggest that there is a strong relationship between design interpretation and the production of strokes. For example, consider the sketches produced in this experiment by two industrial designers shown in Figure 3(a) and Figure 3(b).

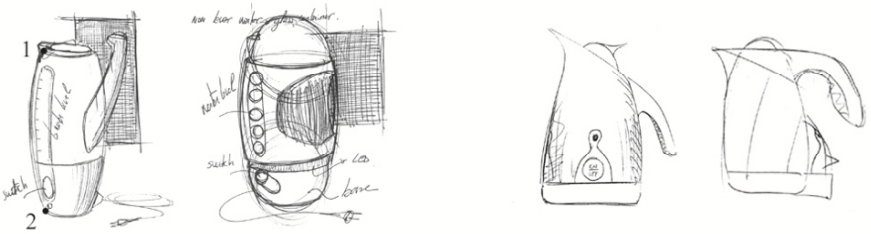


Figure 3. Two sequences of sketches produced by two participants.

The sketches are presented in the sequence they were produced, that is, the sketches illustrated on the right of each pair were produced immediately

after the sketches on their left. Figure 3(a) shows that, initially, the participant produced the outline of the body and the base of the kettle by continuous strokes, that is, from point 1 to point 2, or vice versa. In the subsequent sketch, the participant drew the base and body by separate strokes. This decomposition between body and base opened to the participant a new range of alternatives. The example in Figure 3(b) shows that, initially, the participant constructed the spout and body by the same stroke, and in the subsequent sketch, the spout was produced independently from the body. Again, this suggests that the participant changed their initial interpretation of the concept; and therefore the structure was also changed. This example shows that small variations in the outlines of the design may stimulate significant changes in its structure. Note that in several sketches produced after Figure 3(a) and Figure 3(b) (not illustrated here), the body, base and spout were repeatedly produced by separate strokes. That is, the structure was maintained.

Similar examples occurred in the sketches produced by other participants. Generally, although not always, changes in the production of strokes occur at intersection points; for example, between the spout and body, handle and body, spout and lid. The decomposition or grouping of elements appears to influence the way subsequent ideas are developed. Once participants had visually decomposed their sketches into a particular set of elements, these decompositions were kept while vertical transformations were performed. Generally, a change of interpretation leads to a lateral transformation where the design is reframed and a new range of alternatives is originated.

Lateral transformations and vertical transformations cannot always be identified by observing shape modifications; sometimes it is necessary to involve the designer's interpretation. For example, Figure 4(a) shows a concept design on the left side of the arrow, and its modification on the right side.

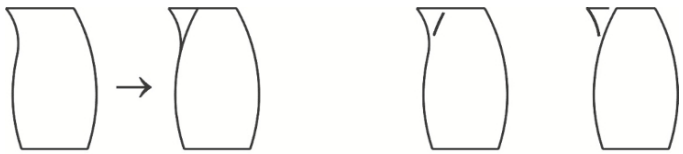


Figure 4. (a) A shape modification, (b) modification interpreted as a vertical transformation, (c) modification interpreted as a lateral transformation.

This can be both a lateral transformation and a vertical transformation. If the interpretation, Figure 4(b), is that a small line has been added to the original concept design then it is a vertical transformation because the new line is considered as an insertion of detail to the original idea. However, if

the interpretation Figure 4(c) is that the added line is an extension of the body's contour then it is a lateral transformation because this movement leads to a slightly different idea compared to the original version. Observe that the original concept design cannot be considered symmetric, and this lack of symmetry is inherited by Figure 4(b). However, the interpretation in Figure 4(c) can be seen as symmetric because the spout becomes a detached element from the symmetric body. Both designs, shown in Figure 4(b) and Figure 4(c), are interpreted as the composition of the same parts; body and spout. However, reinterpretation of shapes often leads to the discovering of emergent parts. While emergent shapes are detected because of a reinterpretation of the design, not all reinterpreted designs lead to emergence. In Figure 4 there are no new emergent parts in addition to the body and spout.

3.2.2. Emergence

Designers often perceive emergent features in their sketches that may not have been initially intended. In this experiment shapes emerged from both interpretative processes where emergent shapes are embedded in the outlines of the design, and transformational processes where emergent shapes are visually suggested by the outlines of the design but they are not graphically represented (Soufi and Edmonds 1996). Consider Figure 5, for example, where the top row shows sketches generated by one of the participants and the second row shows schematic representations used as explanatory illustrations.

The sketch in Figure 5(a) may be perceived as a composition of two elements, as illustrated in the schematic representation. In the subsequent sketch, Figure 5(b), perhaps because the designer focused on functional aspects such as the introduction of a lid on the top part of the kettle, a new element emerged. This suggests that the central line of the initial concept has been extended in order to respond to an emergent interpretation. The thick line in the schematic outlines the emergent shape.

The subsequent sketch in Figure 5(c), suggests again the emergence of a new element. The semi-circular shape on the top of the kettle is now replaced with a complete circle. In sketch shown in Figure 5(d), the designer reinterprets an element that was initially perceived, but which disappeared during the process. The schematic shows the re-emerged element.

This example shows how designers take advantage of emergent shapes, especially from transformational processes. This suggests that not always the structure depends on visual elements but sometimes elements constructed from imagery may also influence the structure.

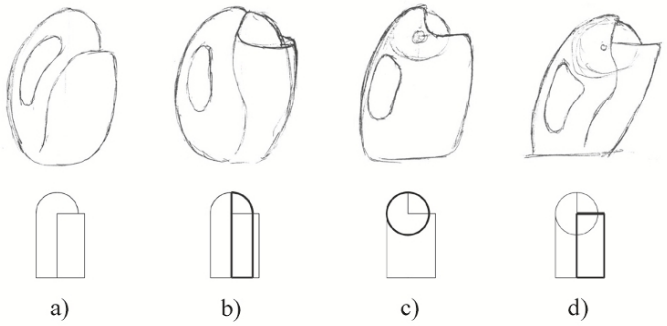


Figure 5. (Top row) Sequence of sketches, (second row) schematic representations of the sketches which highlight emergent features.

3.2.3. Levels of abstraction

Ambiguous and vaguely detailed sketches are not only employed in the preliminary design phases, but they also in refinement phases. Once designers obtain a promising and detailed concept design, they often step back to higher levels of abstraction in order to explore and evaluate the idea from its essence, and omitting irrelevant constraints. Liu et al. (2003) discuss three levels of abstraction, namely topological solution, spatial configuration and physical embodiment levels. While in the first and second levels concept designs are represented with diagrams such as 'bubble' charts, in the physical embodiment level, concept designs are represented in terms of shapes. Although the three levels are related this paper only deals with physical embodiment levels of abstraction.

Designers perceive and generate sketches at different levels of complexity. Generally, there is a correlation between the level of abstraction and the complexity of sketches. The lower the complexity, the higher the level of abstraction, and vice versa (McGown and Green 1998).

In the experiment is observed that most participants progressed with an oscillating search approach, where the complexity of the sketches fluctuates according to the aspects of the design being considered at each moment. Consider, for example, a sequence of sketches generated by one participant shown in Figure 7 in the order they were produced from left to right. Overall the sequence of sketches shows an oscillating exploration process, in terms of complexity/abstraction. Although the complexity of sketches varied through the design process the structure appears to be kept, even when abstract representations suggest alternative structures. In such cases structure seems to prevail above outlines.

However, not always a single sketch is used to represent a particular abstraction, but often different levels of abstraction may be explored in the same sketch. For example, the use of grids, regulating lines and other types

of guide lines are often used by designers to attend higher levels of abstraction. The use of such complementary shapes is not always represented but sometimes they are constructed, perhaps unconsciously, by the mind. Guide lines serve to establish and explain the structure of the design since they order relationships and control the placement, size and proportions of selected elements (Ching 1998). Figure 7 illustrates a sequence of sketches, produced by one participant, which suggest that guide lines were used during the exploration process.

Figure 6. Exploring designs at different levels of abstraction.

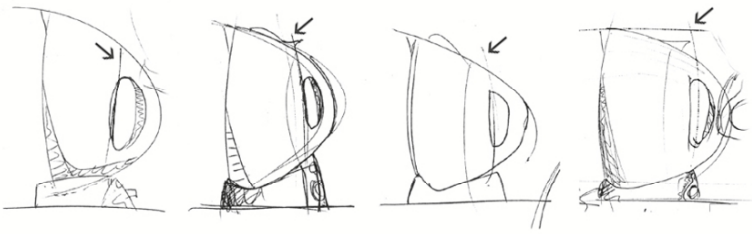


Figure 7. Exploring designs by using guide lines.

Observe that some strokes (pointed with and arrow) do not seem to be part of the concept design, but they are complementary lines that assist the designer in defining the handle of the kettle. Since guide lines are employed to establish relationships between elements they are part of the structural composition. These guide lines facilitates the construction of symmetrical kettles, which appears to be the intention of the designer.

Kolarevic (1997) argues that complementary lines become more interesting when they are not only used as rigid skeletons for the construction of design alternatives, but they may also be dynamic. In other words, manipulations of guide lines assist in exploring design alternatives.

4. A Formal Process for Exploring Designs

This section describes how shape rules of various forms can model the cognitive processes of interpretation, emergence and abstraction associated with exploration with sketches at early design stages. To do this an abstract example is used of generating shapes. These are aligned strongly with the

kind of product sketch observed in section 3 and illustrate how the various shape rules work without being tied to a particular product and its associations. In Section 5 shape rules are formulated for some of the exploratory processes on the jug kettle described in Section 3.

4.1. EXPLORING DESIGNS

In the exploration process the transformation from one concept design into another can be in myriad different ways. In order to guide the exploration process, designers frame the possibilities according to their design intentions and interpretations. Shapes are decomposed as a means to model designer's personal interpretation of sketches, with further shape rules to model designer's intentions.

This model suggests potential areas for computer support in exploring concept designs. These include the application of generative shape descriptions to explore sequences of designs that are consistent with designers' perceptions. The model also provides insights into the mechanisms of exploring product designs.

The examination of a particular design space is often unachievable because the number of alternatives to consider may be impossibly large. One way of exploring design spaces is through generation of design families within the space. That is, sequences of designs that accomplish certain criteria. For example consider Figure 8 as an initial concept design.

Figure 8. Initial concept design.

This initial shape is open to a wide choice and interpretations. For instance, the relationship between the circle and the outer outline may be interpreted as shows the rule 1 in Figure 9. A circle is added to the design whenever three connected arcs –with certain conditions– are found. Note that both sides of the rule are parametric. The concern of how parameters are described is not within the scope of this paper.

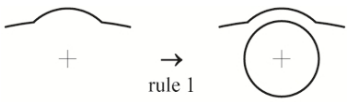


Figure 9. Shape rule for inserting a circle.

While some elements are constructed in relation to current outlines, as shown in rule 1, other elements may be constructed in relation to the supportive shapes, such as the structure. For example, the remaining outlines of the initial shape may be formalized by placing decomposition points and decomposition lines as in Figure 10.

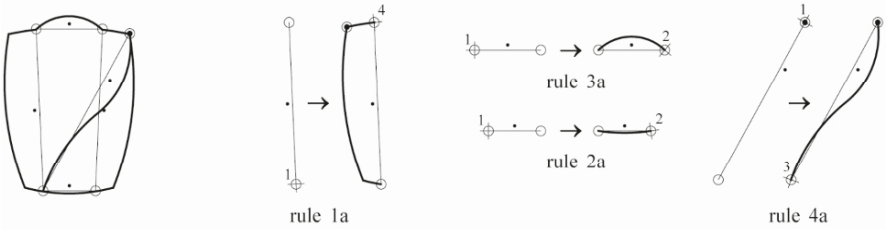


Figure 10. Diagram of elements and decomposition rules define the initial design.

The extremities of each perceived element are identified with breaking points or decomposition points. Each element can be represented by a decomposition line, which joins its two extremities. Decomposition lines are non-terminal shapes that assist the formulation of the shape rules but are not part of the final design. This shape or diagram of elements represented by the decomposition lines is closely related to the perceived structure, and even sometimes the diagram of elements and structure may correspond.

The initial shape is decomposed into five elements of which two are defined by the same rule. Adjacent to the decomposition lines are labels that ensure that each rule is applied in the right place and in the right position. Observe that one decomposition point in rule 4(a) is labeled, indicating that this point is attached to the outline in rule 1a. Thus rule 4(a) relies upon rule 1a. New designs are generated by manipulating the parameters of the rules, that is, the radius of the arcs are modified as well as the lengths and angles between arcs in rule 3(a) and rule 4(a). A sequence of designs is shown in Figure 11.

The parameters can be randomly modified according to constrains defined by designers. Although the rules that modify the outlines are not considered in this paper some types are suggested by Prats et al. (2004). In order to make the generative process more understandable the rules are applied one at a time. For example, once the design has been reconstructed by rules 1(a)-4(a) and rule 1, Figure 11(b), the outlines of each rule can be modified, Figure 11(c)-Figure 11(j). Note that rule 1a is applied twice in the design and therefore rule 4(a) has two possible solutions since it relies upon rule 1, Figure 11(e).

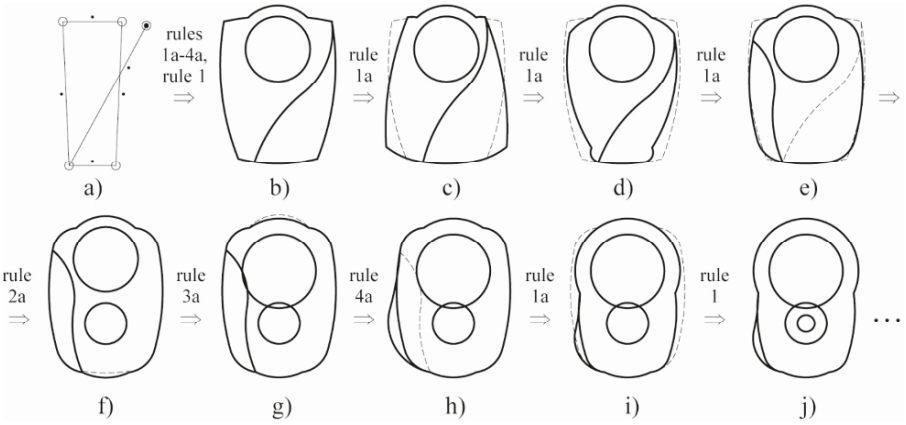


Figure 11. (a) Diagram of elements assigned in Figure 10, (b)-(j) manipulations of the outlines through decomposition rules.

It is interesting to observe that small variations on the outlines produce significant perceptual consequences to the initial concept design. These variations assist explorations of designs that are consistent with the interpretation formalized. Each generated design in a family preserves the original diagram of elements, or structure. The generated designs maintain a designer's 'frame' even if outlines are modified randomly by computers.

4.3.1. Reinterpretation

During the exploration process it is possible to obtain many different designs whilst preserving a particular interpretation of the design. Each design alternative emerges after application of vertical transformations to the concept design. This type of transformations is crucial when designers want to explore versions of a chosen concept design while its essence is kept. However, as seen in the experiment, changes on the interpretation are also crucial in exploration stages. This assists designers in reframing the design space which leads to consider new design alternatives that previously were not taken into account. Consider for example Figure 12. This new decomposition can generate designs that were unachievable through the previous decomposition.

4.3.2. Emergence

In our empirical study it has been examined how unexpected shapes emerge in the designer's eye during the conceptual stages of design. Stiny (1980) has proposed a method that supports computation of emergent shapes. He argues that shapes do not have finite numbers of parts and therefore can be freely decomposed. Thus emergent shapes can be recognized at any stage of the computation. The history of emergence and formal devices for

computing with emergence and ambiguity are discussed in (Knight 2003a; Knight 2003b). Emergent shapes can appear in two ways: (i) application of a defined rule in an unexpected place, and (ii) the designer defines a new rule after discovering an emergent shape. The examples in Figure 13 show how the defined rule 1 can be applied to unexpected places.

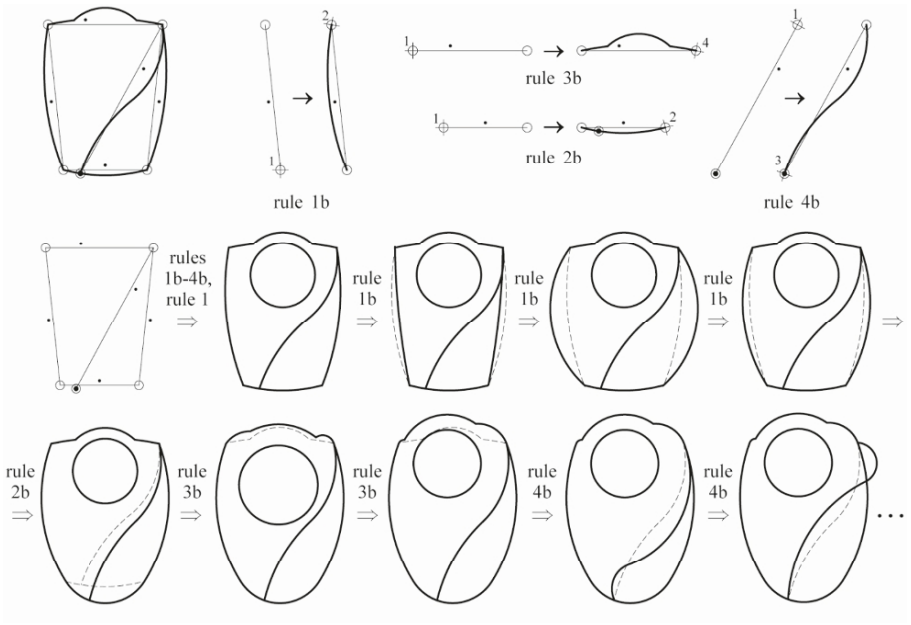


Figure 12. (Top) New diagram of elements and rules define a reinterpretation of the initial concept design (a)-(j) manipulation of outlines.

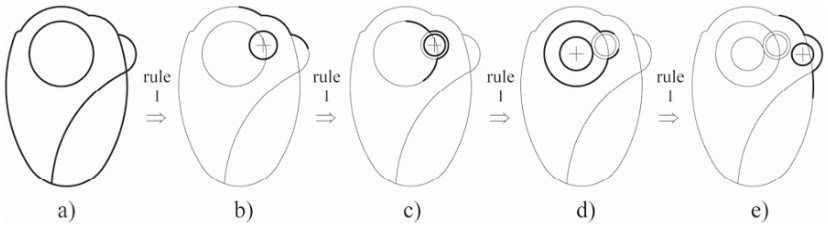


Figure 13. Four different matches of rule 1 on the same design.

4.3.3. Levels of abstraction

One important aspect of the creative process is that shapes can be perceived and represented at different levels of abstraction. During the design process, designers may explore designs at a detailed level by focusing on specific outlines of the shape while temporarily ignoring other outlines. Also, designers may explore designs at a more abstract level by focusing on the

arrangement of the elements perceived in the shape. For example, Figure 14 shows that manipulations in the diagram of elements generate new alternatives of a chosen concept design (in this case Figure 13(d)). Note that each diagram of elements and outline represent the same concept design, but in the left side of each figure the outline is not represented, and in the right side the diagram of elements is not represented.

The transformations of the diagram of elements are performed in order to satisfy design intentions in an overall sense. In Figure 14(b) the diagram of elements is approximated into a golden rectangle, in Figure 14(c) into a square, or in Figure 14(d) into a more dynamic type of shape.

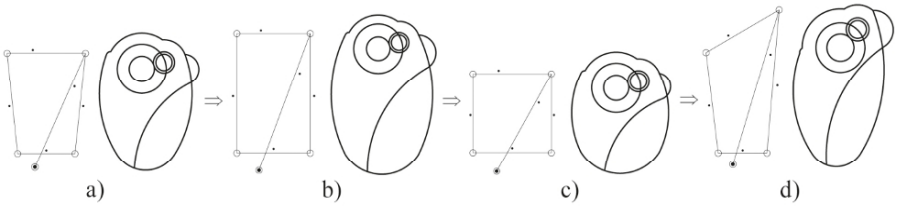


Figure 14. (Left) Manipulations of the diagram of elements in Figure 12, (right) outlines attached to diagram of elements.

Note that once a promising diagram of elements has been found all previous designs –designs in Figure 12 and Figure 13– and also potential designs in the associated design spaces with the same shape rules can be adapted to the transformed diagram of elements. As mentioned earlier, exploration of designs not only consists in manipulating visible outlines, but also examining hidden structures to find internal coherence in designs.

New supportive lines can also be added by defining new rules. For instance, rule x in Figure 15 traces a line between two intersection points. Such lines may be used to relate the diameter of the circle with features of the outline. Rule z expands or contracts the circle in order to meet the supportive line. This type of visual relationships between elements is often used by designers as discussed in Section 3, Figure 7.

5. Discussion

This paper started by examining the role of visual perception in design exploration. Our examination has focused on three perceptual processes which appear to be essential in the exploration of new designs; reinterpretation, emergence and abstraction. We propose that assigning particular structures to designs assists exploration of new designs. In addition, structures may ensure that computationally generated designs are consistent with the designer's perceptual processes. Structures are defined according to designer's perceptions and intentions. Consider for example the

sketch in Figure 16(a), which has been taken from our empirical study, Figure 3(b).

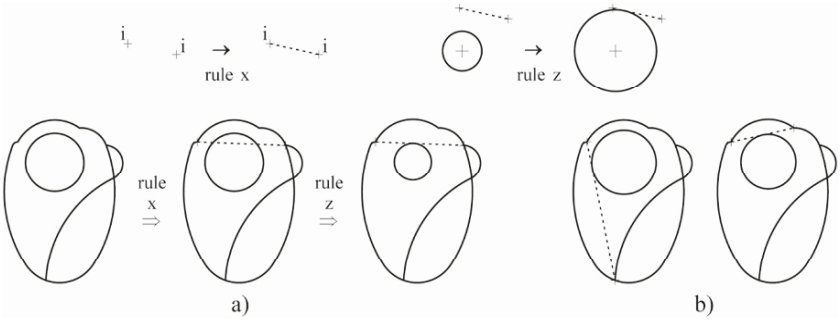


Figure 15. (a) Application of rules x and z establish visual relationships between elements, (b) two different alternatives.

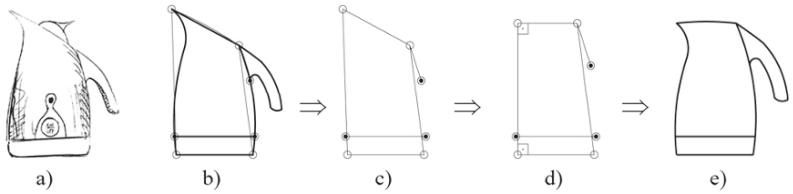


Figure 16. Manipulation of a design.

According to the process presented in this paper, the design can be decomposed by assigning decomposition points and decomposition lines as shown in Figure 16(b). The added lines take the form of a structure, Figure 16(c), which can be manipulated according to aesthetic preferences. A rule (not illustrated here) that arranges two connected lines into a right angle generates the structure shown in Figure 16(d). This is just one possibility from a range of configurations, as previously illustrated in Figure 14. Figure 16(e) shows the outlines attached to the modified structure.

New elements of detail may be added to this design as shown in Figure 17(a). If the introduction of these elements is defined in terms of shape rules they may generate additional designs as previously illustrated in Figure 13.

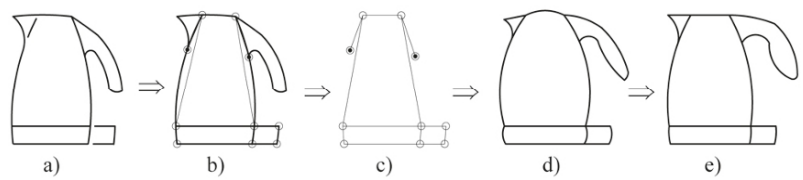


Figure 17. Reinterpretation of the design in Figure 16 and manipulations of outlines.

Once the elements are in place an inspection of the design may suggest new interpretations. Figure 17(c) shows a possible structure defined according to a new reinterpretation. Design alternatives can be explored by modifying the outlines defined by the structure. Figure 17(d) and Figure 17(e) show two examples.

With the purpose of inserting a lid in the kettle a new rule could be defined. For example, two symmetrical curves are found (shown in thick line in Figure 18(a)) and they are joined with an arc from their end points as shows Figure 18(b). However, this rule can find more instances in the design which generate unexpected designs that may provide emergent features as shows Figure 18(d). Observe the similarities between this design and the sketch in Figure 18(e) taken from our empirical study.

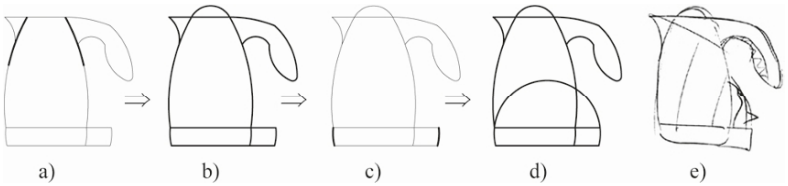


Figure 18. Insertion and emergence of new features.

This example attempts to show that sequences of designs, at least in convergent thinking, can be traced in a systematic and logical way. Here we have traced a path that formalizes the sequence of modifying one sketch, Figure 16(a), into another, Figure 18(e). This path has been constructed by means of using reinterpretation, emergence and abstraction as examined in our empirical study. In the experiment the designer produced these sketches in one single step, but the parallelism between imagery and perceptual processes discussed by Kosslyn (1990) bring us to the hypothesis that the designer followed a mental process comparable to the path shown in Figure 16-Figure 18. The evidence of the sketches suggests that the cognitive processes of reinterpretation, emergence and abstraction are widely used. These are expressed in terms of shape rules in an associated model. Further work is being undertaken in implementing the kinds of shape rule on curved shapes that are required for product design.

Acknowledgements

The authors thank the participants in the experiment and the Open University for its financial support for Miquel Prats.

References

Akin, O: 2001, Variants in design cognition, *in* CM Eastman, WM McCracken and WC Newstetter (eds), *Design Knowing and Learning: Cognition in Design Education*, Elsevier, pp. 105-124.

Arnheim, R: 1966, *Towards a Psychology of Art*, Faber and Faber, London, UK.

Arnheim, R: 1974, *Art and Visual Perception: A Psychology of the Creative Eye*, University of California Press, Berkeley, CA.

Ching, FDK: 1998, *Design Drawing*, John Wiley, New York, NY.

Cross, N: 1994, *Engineering Design Methods: Strategies for Product Design*, John Wiley, Chichester, UK.

Do, EY and Gross, MD: 1996, Drawing as a means to design reasoning, *Artificial Intelligence in Design,* Palo Alto, CA.

Ferguson, ES: 1992, *Engineering and the Mind's Eye*, MIT Press, Cambridge, MA.

Gero, JS: 1996, Creativity, emergence and evolution in design, *Knowledge-Based Systems* **9**(7): 435-444.

Goel, V: 1995, *Sketches of Thought*, MIT Press, Cambridge, MA.

Goldschmidt, G: 1994, On visual design thinking: The vis kids of architecture, *Design Studies* **15**(2): 158-174.

Hoover, SP and Rinderle, JR: 1991, Models and abstractions in design, *Design Studies* **12**(4): 237-245.

Kepes, G: 1944, *Language of Vision*, P Theobald, Chicago, IL.

Knight, TW: 2003a, Computing with emergence, *Environment and Planning B: Planning and Design* **30**: 125-155.

Knight, TW: 2003b, Computing with ambiguity, *Environment and Planning B: Planning and Design* **30**: 165-180.

Kolarevic, B: 1997, Designing with regulating lines and geometric relations, *IDATER97*, Loughborough University, UK.

Kosslyn, SM: 1990, Mental imagery, *in* DN Osherson, SM Kosslyn and JM Hollerbach (eds), *Visual Cognition and Action*, The MIT Press, London, England **2**: 73-97.

Lawson, BR: 2004, *What Designers Know*, Architectural Press, London.

Liu, YC and Bligh, T: 2003, Towards and 'ideal' approach for concept generation, *Design Studies* **24**(4): 341-355.

McCormack, J and Cagan, J: 2004, Speaking the Buick language: Capturing, understanding, and exploring brand identity with shape grammars, *Design Studies* **25**(1): 1-29.

McGown, A and Green, G: 1998, Visible ideas: Information patterns of conceptual sketch activity, *Design Studies* **19**(4): 431-453.

Prats, M and Jowers, I: 2004, Improving product design via a shape grammar tool, *8th International Design Conference (Design 2004)*, Dubrovnik, Croatia.

Purcell, T and Gero, JS: 1998, Drawings and the design process, *Design Studies* **19**(4): 389-430.

Schön, DA: 1988, Designing: Rules, types and worlds, *Design Studies* **9**(3): 181-190.

Schön, DA and Wiggins, G: 1992, Kinds of seeing and their functions in designing, *Design Studies* **13**(2): 135-156.

Smithers, T: 2001, Is sketching an aid to memory or a kind of thinking?, *in* JS Gero, B Tversky and T Purcell (eds), *Visual and Spatial Reasoning in Design II*, Key Centre of Design Computing and Cognition, University of Sydney, Australia**:** 165-176.

Soufi, B and Edmonds, E: 1996, The cognitive basis of emergence: Implications for design support, *Design Studies* **17**(4): 451-463.

Stacey, M: 2005, Psychological challenges for the analysis of style, Leicester, De Montfort University**:** *Internal Report.*

Stiny, G: 1980, Introduction to shape and shape grammars, *Environment and Planning B: Planning and Design* **7**(3): 343-351.

Stiny, G: 2006, *Shape: Talking about Seeing and Doing*, MIT Press, Cambridge, MA.

Stiny, G and Gips, J: 1972, Shape grammars and the generative specification of painting and sculpture, *Proceedings of IFIP Congress 71*, Amsterdam: North-Holland, pp. 1460-1465.

Suwa, M: 2005, Differentiation: Designers are more than being good at designing, *in* JS Gero and N Bonnardel (eds), *Studying Designers '05*, Key Centre of Design Computing and Cognition, University of Sydney, pp. 33-38.

Suwa, M and Gero, JS: 1999, Unexpected discoveries: How designers discover hidden features in sketches, *in* JS Gero and B Tversky (eds), *Visual and Spatial Reasoning in Design*, Key Centre of Design Computing and Cognition, University of Sydney, Australia, pp. 145-162.

Suwa, M and Tversky, B: 2003, Constructive perception: A metacognitive skill for coordinating perception and conception, *Proceedings of the Cognitive Science Society Meetings*.

Tversky, B: 2001, Spatial schemas in depictions, *in* M Gattis (eds), *Spatial Schemas and Abstract Thought*, MIT Press, Cambridge, pp. 79-111.

Van Sommers, P: 1984, *Drawing and Cognition*, Cambridge University Press, Cambridge.

DIGITAL SKETCHING IN A MULTI-ACTOR ENVIRONMENT

ALEXANDER KOUTAMANIS
Delft University of Technology, The Netherlands

Abstract. The paper discusses digital sketching in the framework of multi-actor design processes. The discussion focuses on the registration, analysis and feedback of annotations made with digital pens on prints of CAD drawings and early stage sketching, both in synchronous and asynchronous situations. It is proposed that the main advantages of this form of digital sketching are the registration of syntagmatic information and the ability to distinguish between different actors. This makes it possible to identify meaningful entities and clarify issues of common authorship or emergence.

1. Digital Sketching and Multi-actor Design Processes

Sketching is one of the means architecture has been successfully employing for the registration and processing of design information. The success of sketching is based on the immediacy, informality and familiarity of sketching procedures and representations, as well as the flexibility and adaptability of means involved (i.e. using practically any drawing implements available on any medium). The main problems of sketching lie in the parsing and disambiguation of its typically dense and multilayered products, especially when they involve more than one sketcher or longer periods of time. The extensive use of video capture in protocol analyses of design situations is indicative of the complexity and tenacity of these tasks. It is arguably for such reasons that digital sketching has yet to come of age. Affordable digital means cannot capture the flexibility, adaptability and mechanical feedback of analogue sketching tools, while the interpretation and dissemination of information in a sketch cannot rely on the exchange structures and information standards used for drawings and models.

Sketching studies tend to focus on the generative and the representational, i.e. the processes (and products) of form generation and the depiction of real or realistic scenes. This paper concentrates on different applications where sketching plays an equally important role. We examine digital sketching in the context of multi-actor design processes. Situations where several parties are actively involved in a process, taking decisions,

J.S. Gero (ed.), Design Computing and Cognition '06, 103–121.
© 2006 *Springer. Printed in the Netherlands.*

creating and amending forms, exploring complementary issues and frequently sharing the same media and representations put additional demands on the interpretation of sketches and their transformation into other representations. Moreover, with the proliferation of digital information in all aspects of professional and private life and the ubiquity of mobile information processing we expect that group processes and multi-actor design environments will become increasingly popular and feasible. Our exploration of digital sketching in these processes and environments also focuses on sketches as annotations of existing design representations. The main reason is that such annotations are quite common and frequently confusing, especially when several actors representing a variety of aspects or viewpoints and contributing overlapping parts of the design are involved.

The following figures present an example that illustrates the problems of group annotations. Figure 1 is a detail from a drawing used during a discussion on the refurbishment of a university building. The seven participants used several media, colors and symbols to put their ideas on paper. The registration of decisions, alternatives and variations varied from inconsistent to chaotic. When the participants met again one week later, reconstruction of the discussion and its conclusions on the basis of the drawings was expectedly biased and tainted by individual viewpoints and opinions. The participants' recollection of actual decisions and conclusions was hazy and inconsistent. One striking example of accidental emergence concerns the coffee machine niche –indicated in the initial proposal, Figure 2, by a horizontal square bracket ([)– that should be placed somewhere in the circulation/reception area. The participants had different ideas concerning the location of this facility and its relations to other areas (e.g. waiting area, information corner etc.). Many proposed alternatives involved rotations and reflections of the bracket form. The superimposition of these alternatives produced accidentally what appears to be a square form in the most popular place for the coffee-machine niche, Figure 3. When asked about this square one week after the meeting, two participants could not recollect what it signified and another two were convinced that it was a cubicle with an unknown function and an unknown originator but nevertheless a useful feature for presentation and orientation purposes.

Digital information processing should resolve such problems by identifying different elements, contributions and intentions. However, it is not always possible to record who did what and when (let alone why). Even with advanced registration systems (including versioning control), many discussions and changes are soon forgotten, reduced to their final state (no history) or even fused together in unexplainable cases of accidental emergence. In this respect digital information processing has still much in common with analogue situations, both in synchronous and asynchronous

modes. As the discreteness of information items remains largely unrelated to user input, we lack the means necessary for parsing digital sketches.

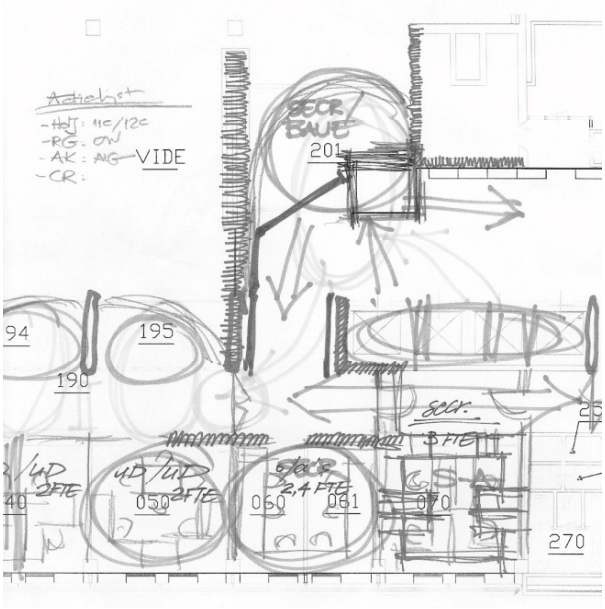


Figure 1. Annotated print of CAD drawing (detail).

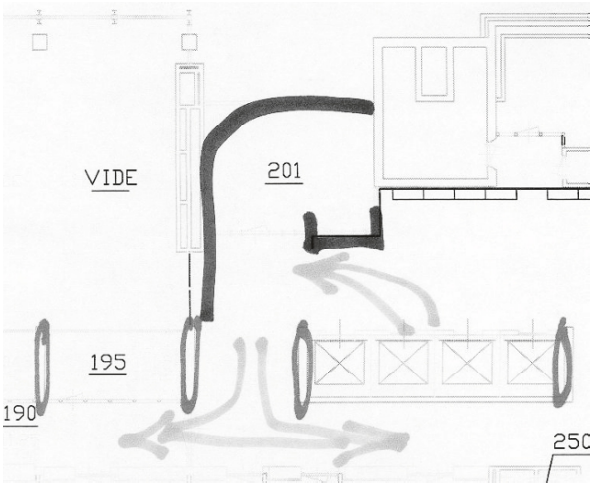


Figure 2. Initial proposal with bracket-shape coffee-machine niche in the middle.

Figure 3. Detail of Figure 1 with accidental square in the middle.

2. Sketching Dimensions and Requirements

Sketching can be analyzed in several dimensions, which it shares with similar activities such as drawing and writing (Van Sommers 1984). The most important are:

1. The *mechanical* dimension: the physical interaction between the sketcher's anatomy and pen, paper, the writing surface etc.
2. The *paradigmatic* dimension: the strokes made by the sketcher and the graphic primitives or symbols they comprise
3. The *syntagmatic* dimension: the sequence by which the drawing is formed by discrete strokes and symbols

The paradigmatic dimension is a traditional focus of computational sketching studies (Achten and Jessurun 2002; Do 2002; Jozen et al. 1999; Koutamanis 2001; Sugishita et al. 1995). The mechanical dimension is largely unexplored in architectural research, with the possible exception of interfaces to 3D environments (Achten et al. 2000; Do 2001; Woessner et al. 2004). The same applies to the syntagmatic dimension, despite attempts to improve the means of protocol analysis and some interesting research in the semantic and syntactical significance of this dimension (Cheng 2004; Gross 1995). In our research the paradigmatic dimension is only a secondary point

of attention. The mechanical dimension is considered in more detail, especially with respect to the comparison between digital and analogue media. The syntagmatic dimension is central to the research, as a primary means for disentangling actions, symbols and individual contributions.

A brief analysis of these dimensions in comparison to current digital technologies for architectural sketching returns several primary limitations:

1. *Viewing limitations*: One class of well-known limitations concerns the viewing distance and angle from computer monitors. Most users agree that viewing information on a standard computer monitor together with more than one other person can be uncomfortable, especially for longer periods and with information that requires attention. The common response to such complaints is to increase the size of the viewing facilities with beamers and large monitors. By comparison analogue documents fare significantly better, allowing very large sizes and multiple views at a low cost.
2. *Interaction limitations*: Even with larger viewing facilities, interaction with the information remains a bottleneck. Normally only one person can manipulate or input information with the computer's keyboard and mouse. The obvious solution is to pass the input devices around, allowing each user present to take a turn. Here again analogue documents are more flexible and adaptable, especially with respect to parallel processes and multiple actors: anyone who can get hold of a pen can interact with any of the available documents (including copies of the same document), frequently simultaneously with others. Group facilities like smartboards and large plasma or LCD touchscreen panels offer a halfway solution (large sizes and multi-actor support but no simultaneous multiple views) limited primarily by high cost.
3. *Tracing and reconstruction*: Digital information allows direct and generally effortless transformation but it is not always possible to record the actors involved and the context of their actions. Document management and versioning systems generally describe document states based on arbitrary time scales and prescriptive process models. In many respects such systems reproduce analogue practices and may even fail to connect to information standards so as to identify the evolution of different aspects in a design and their interrelationships.

If we ignore the differences between digital and analog, small and large, expensive and affordable, we can compile a number of requirements for the flexible, adaptable, reliable and direct processing of design and building information that covers sketching, drawing and related activities:

- *Large viewing facilities*: size limitations are unacceptable to a profession accustomed to A0 and A1 sheets. We may be getting

used to the significantly lower resolution of computer monitors but lack of overview is a common complaint.

- *Multiple copies*: using multiple copies of the same document supports the exploration and comparison of variations and alternatives, including fast backtracking and parallel development of design solutions.
- *Free interaction* with all documents including overlaying of different documents, direct modification and markup, as well as allowing many actors to work together on one document.
- *The ability to distinguish between different actors and actions:* analysis of the paradigmatic and the syntagmatic dimensions so as to register and record complex situations, untangle contributions, disambiguate forms and interpret intentions in a reliable and consistent manner.

3. Analogue to Digital, Digital to Analog

The proliferation of digital information in all aspects of daily life, from leisure and entertainment to professional activities, is already an established fact. In most situations we assume that up-to-date, interactive information is readily available through a variety of digital media, normally at a relatively low cost. Moreover, the places where we access and consume information are becoming quite diffuse due to the recent increase of (multi)media devices at home and the popularization of mobile information processing. Still, we often transfer digital information to analogue carriers and from there back to the computer. Such transitions from analogue to digital information and vice versa are commonplace in architectural computing for a number of reasons, including:

- Ergonomic limitations of digital media, especially in comparison to analogue ones
- The distributed structure of the building industry
- The relatively low level of computerization in architecture and building

These transitions constitute ultimately a cyclical process, Figure 4. Feedback to earlier stages in this process permits comparison between different states of a design and hence facilitates accurate and precise identification of changes, as well as possible causes and conflicts. In this framework the most demanding type of transition (and hence the departure for our research) is the one from an analogue image produced from a digital model back to the computer. This may sound rather convoluted but in fact it represents an everyday need in architectural computerization. For the largest part of the field's history CAAD has been trying to replace analogue design processes and products with digital ones. However, computerization of

architectural and building practice has produced results that are markedly different from many of the underlying assumptions of CAAD. One clear contrast with the intentions of digital design is the increase in paper consumption in the design office. Similarly to office automation, drawing, modeling and visualization systems appear to aim at the production of paper documents. Even though digital information is exchanged and processed more than ever, prints and plots remain the basis of building specification and design communication. Attempts to replace the paper carrier with digital media (from laptops and palmtops on the building site to smartboards in design presentations and meetings) have given us glimpses of the possible future but have yet to supplant analogue documentation.

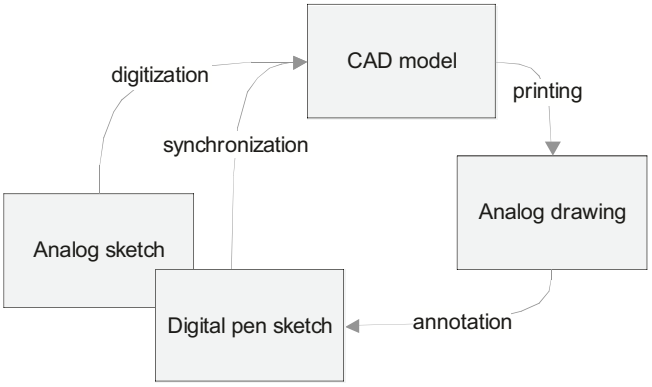


Figure 4. Analog-to-digital cycle.

The transition from original analogue image to the computer mostly takes place by means of manual and optical digitizers. Manual digitization is based on the combination of graphics tablets with pen-like pointing devices and is characterized by low user threshold, as the mechanical and ergonomic properties of such devices are similar to these of familiar analogue media: an action that would produce a stroke on paper is also captured by a manual digitizer and produces a similar result. Arguably more important for the transfer of analogue images to the computer than mechanical scanners have been optical digitizers (scanners). In terms of output the essential difference between manual and optical digitizers is that scanners produce a pixel image. This makes optical digitization less attractive for a number of applications but in terms of utility scanners provide more flexibility, tolerance and ease. They accept a wide variety of analogue documents and hence allow the user to make full use of analogue skills. In doing so, they compensate for the weaknesses of manual digitizers in mechanical aspects. Manual digitizers also exhibit limitations in terms of cognitive ergonomics. For example,

sketching with most tablets involves constantly looking away from the hand and frequent interruptions in order to give commands to the computer. Admittedly experienced users may cross over to "blind drawing" but, as with blind typing, this involves weakening of significant forms of mechanical and perceptual feedback.

Feedback from hand-drawn annotations on prints and plots is generally made by hand in CAD programs. There are good technical reasons for not digitizing such complex images. Distinguishing between old (i.e. printed) information and new (annotations) in the products of optical digitizers is a cumbersome and expensive task, beyond the reach of most automated recognition systems and rather inefficient to perform interactively on the basis of a rudimentary vectorization. With manual digitizers a similar interactive transfer is possible: the user distinguishes between old and new information and draw the new one in the CAD system (preferably using overlay facilities of the tablet), either as annotations or directly as changes in the existing graphic entities. This obviously duplicates the time and effort required for making the annotations. More crucially it involves human interpretation of the annotations and consequently the possibility of errors.

An alternative to redrawing is direct interaction with the digital information, ranging from direct modification to markup and whiteboarding. Most interaction facilities that go beyond the keyboard and the mouse are based on combinations of monitors with mechanical digitizers so as to recreate the mechanical and perceptual experience of analogue drawing and writing. They include facilities for individual use such as LCD tablets, tablet PCs or palmtop devices, and group facilities such as smartboards and interactive touchscreen additions to plasma and large LCD monitors. From a technical viewpoint most of the smaller systems use pressure-sensitive screens, while the larger ones employ infrared scanning to determine the position of a pointer on the projection surface. The alignment problems that used to plague such systems are no longer an issue but cost may still be an objection, especially with the larger solutions.

4. Implementation

A comparison of the viewing requirements stated above with the capabilities of analogue documents suggests that the prints and plots routinely produced from digital representations could meet our expectations, provided that the interaction with these analogue documents would return reliable analyses of user input and focused feedback to the digital representations. This interaction should be characterized by a higher degree of mobility than current design automation: information processing should be brought to the drawing and not the other way around. A new technology that could satisfy our requirements is the *digital pen*. The name refers to a number of recent

devices and ideas that attempt to incorporate digital processing capabilities to the familiar processes of writing and drawing on paper. The currently most advanced / established among these is the Anoto digital pen and paper (http://www.anoto.com). This technology is actually a combination of mechanical and optical digitization. Anoto pens can write on any kind of paper form that is covered with a proprietary dot pattern with a nominal spacing of 0.3 mm. A minute portion of the total pattern uniquely defines the absolute position in the form. A number of custom symbols for specific actions (e.g. changing stroke color or thickness) can also be included on the form. The digital pen is equipped with a tiny infrared LED camera of the CMOS image-sensing type. The camera is positioned beside the ballpoint tip and takes 50-100 digital snapshots of strokes made by the pen on the dot pattern within a 7 mm range from the tip. The snapshots are stored in the pen as a series of map coordinates that correspond to the exact location of the strokes (as continuous curves) on the particular page. In fact the camera (being sensitive to infrared) does not capture the ink traces on the paper but the movement of the pen with respect to the dot pattern.

The drawings are transferred (synchronized) to a computer using Bluetooth or USB. The digital images produced by synchronization are precise and exact copies of the analogue drawings. In transferring the images the pen also reports on which form the drawing has been made. This automatic document management allows users to switch between different documents without having to keep track of the changes. Post-processing includes grouping of strokes into higher-level primitives (e.g. OCR). The digital images are dynamic and can play back the sequence of strokes made on paper. For our purposes this technology has two main advantages:

1. *Feedback from analogue to digital*: using a digital pen on a print of a CAD drawing records only the new information (annotations) and transfers it back to the computer. As such it closes the analog-to-digital cycle in an efficient and unobtrusive way, Figure 4.
2. *Recording of syntagmatic information*: the digital pen captures strokes as discrete events and transfers this information to the computer. Analysis of the syntagmatic dimension contributes to the recognition of symbols (as contiguous series of strokes), the identification of relationships between symbols (in terms of temporal clustering) and the distinction between different actions and decisions (by means of temporal distance).

4. Synchronous and Asynchronous Group Processes

Our exploration of digital sketching in the annotation and modification of CAD drawings took place in both synchronous and asynchronous multi-actor settings. The synchronous side was covered in design meetings

involving three to five participants, each representing a different aspect or specialization (architectural, interior and structural design, building cost, brief satisfaction). Each participant used a separate digital pen so that we could distinguish between individuals / aspects. The drawings used in the design meetings were normal laser prints from CAD models, with the difference was that they were overlaid with the Anoto dot pattern. In the asynchronous settings each participant (again representing a particular aspect or specialization) was also issued with an own set of laser prints that formed the background to a number of parallel, overlapping tasks.

In both settings we focused on the correlation and integration of information from different aspects. In the synchronous cases a large part of this took place in the meetings on the basis of the analogue documents. The main contribution of the digital pen technology was to distinguish between actors, aspects, actions and versions of a decision. The final versions (including histories) were fed back to the computer and linked to relevant design entities. These links formed the departure for modifications in the design. Modifications were frequently guided by the history of the particular decision (e.g. as a means of adding details than may have been omitted from the final version). As the asynchronous situations missed the correlation of aspects that took place in the meetings, the input from different actors was collated in the CAD models and fed back to the actors for a short round of verification, comments and possible adaptations.

Participants experienced few problems with the mechanical aspects of digital sketching. With just a few experimental sketches they became aware of the main limitations of the pen and were able to avoid their consequences. This was also facilitated by the nature of their tasks: making textual and graphic annotations on a drawing is less demanding than making an artistic sketch of a three-dimensional scene. The only persistent irritations were that (a) line thickness and color were visible only in the digital version, and more significantly that (b) strokes made on heavily printed parts of the drawing did not register because e.g. dense hatching interfered with the dot pattern.

The ability to record syntagmatic information meant that we could distinguish clearly between states, actions and actors, Figure 5. In the follow-up meetings, which normally took place one week later, the participants' recollection of events and decisions was refreshed by playing back the sequence of strokes made with each pen. This improved the accuracy of decisions taken by the whole group and facilitated concentrating on their consequences for the development of the design. It also supported continuity in the design process, including backtracking to earlier states and decisions. Syntagmatic information played an important role in the disambiguation of the paradigmatic structure of the typically messy annotated drawings produced in a design meeting. Visual fusing of adjacent or overlapping strokes from the same or different actors in a synchronous

situation, as in the accidental emergence case in Figure 3, was reduced to an absolute minimum (a couple of temporally sequential strokes).

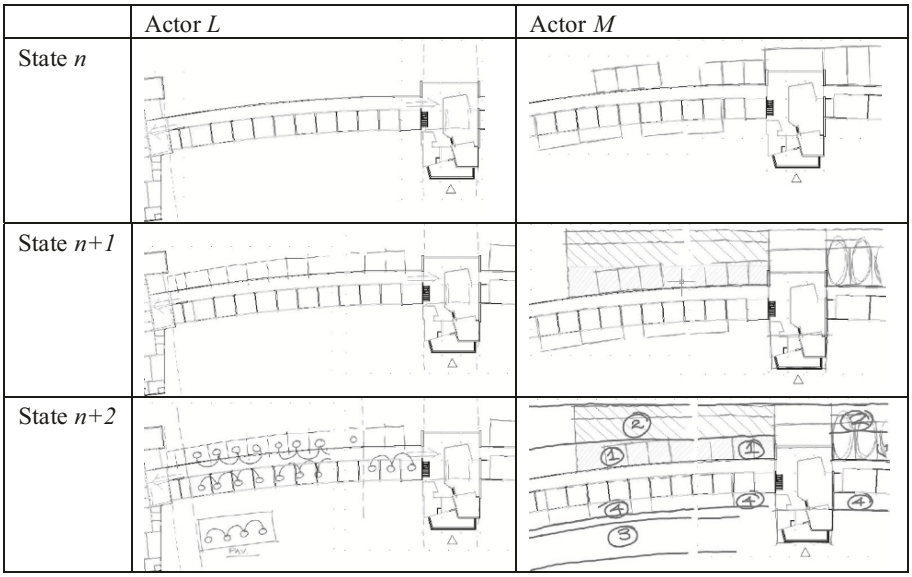


Figure 5. Syntagmatic parsing of synchronous case.

Feedback of annotations to the CAD files used for the production of the prints was assisted by (a) the high precision of sketches, drawings and texts produced with the digital pen, and (b) the built-in document management capabilities. As a result, synchronized information could be directly linked to the appropriate views of a CAD model as an overlaid pixel image, as markup, Figure 6 and Figure 7.

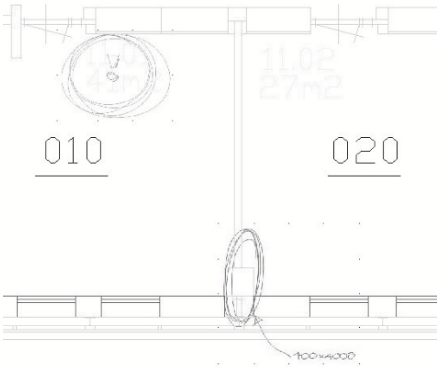


Figure 6. Annotation feedback to CAD.

OCR of verbal annotations enriched post-processing by returning e.g. numerical values for proposed metric modifications and labels that could be matched to entity properties. For example, the text “door” could be linked directly to door symbols in the vicinity of the label or to layers containing doors.

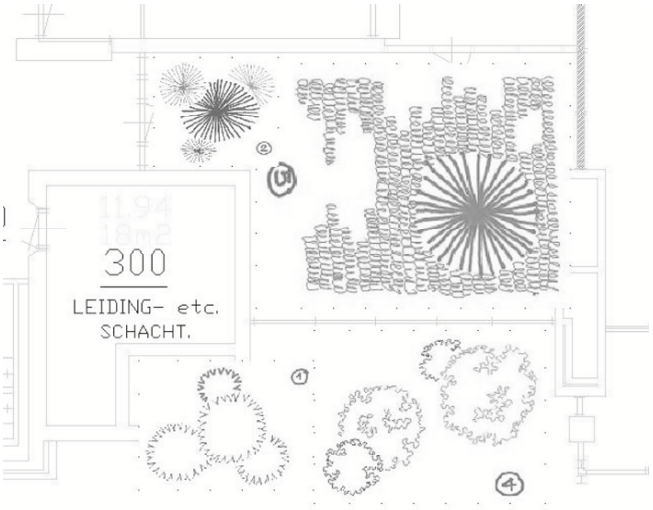


Figure 7. Composite view in CAD produced through annotation feedback.

A further elaboration of the standard synchronization was the automatic derivation of selection areas in the CAD model on the basis of the form and size of annotations. For example, a bubble form drawn with multiple lines triggered a window-type selection with the dimensions of the bounding box of the annotation, Figure 8. These selection areas assist in identifying the relevance of annotations to specific parts of the CAD model, as in most cases the annotations overlap with the elements they refer to. Relevance identification by means of such selection areas was instrumental in the correlation of annotations from different actors in asynchronous situations, as well as for the identification of relevant actors for subsequent communication and development actions and tasks.

The parsing of an image into groups of strokes also opens up possibilities for the recognition of design entities and symbols in a manner similar to OCR (Do 2001; Koutamanis 1995). This was not attempted with the annotations that were fed back to the CAD system. The main reason was that the annotations we handled were too elliptical to present a coherent and consistent basis for identifying symbols. However, we were able to observe that such symbols tended towards the personal and idiosyncratic. This suggests further research into the use of simple strokes to compose a symbol

(paradigmatic dimension), as well as into how these strokes are spread in time (syntagmatic dimension) due to e.g. mechanical issues.

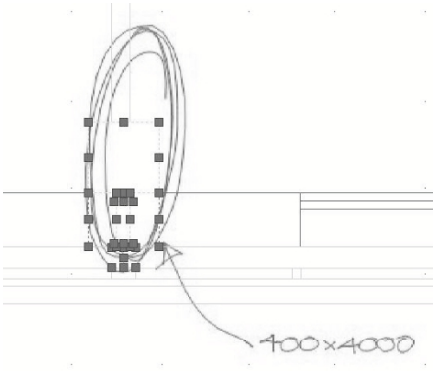


Figure 8. Automatic selection of entities on basis of annotation feedback.

It should be noted that most of the above observations refer to synchronous cases. In the asynchronous setting we were unable to observe objective advantages over other means of interactions with digital information (e.g. markup with a tablet). On the contrary, users preferred to work with a tablet PC and a LCD tablet that were also available. The only clear advantages of the digital pen in the asynchronous cases were mobility and the ability to use large drawings. These advantages were of particular interest in later design stages, when the amount and precision of design information, as well as the underlying history of decision-taking, were much higher and more binding. Tracing back previous states and relevant decisions along the syntagmatic dimension recorded in digital sketching was instrumental in clarifying the constraints of particular situations.

5. Beyond Annotations

A wider exploration of the applicability of digital sketching was performed through the progressive relaxation of the constraints used so far, starting with the feedback to computer documents. In design meetings that started from scratch (as far as visual design documentation is concerned), disambiguation of the final products also benefited from the recorded syntagmatic information, Figure 9. The allocation of a pen to each actor allowed for greater flexibility than with e.g. smartboards. Parsing the images produced in the meetings into discrete actions, decisions and relationships was technically less challenging, as there was no feedback to CAD models.

Probably the most interesting observations made in such settings concerned the use of each other's information as a reference frame or point of

departure. Sketchers referred not only to their own previous input but also to that of the others, frequently in a positive sense.

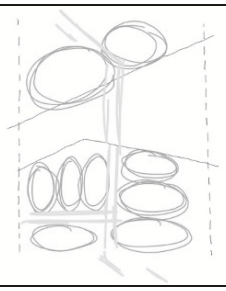
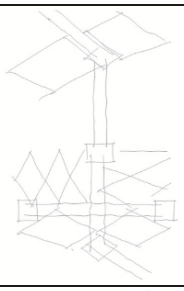
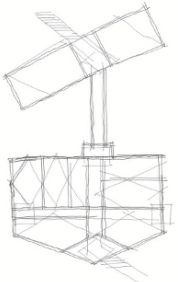
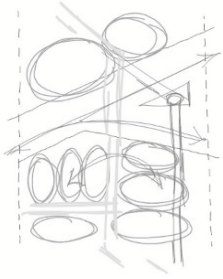
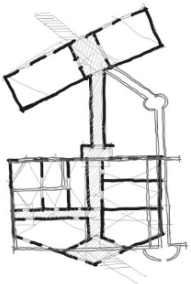
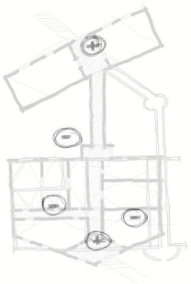
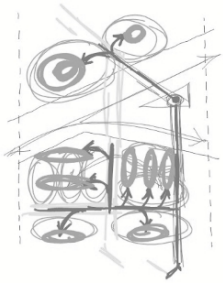
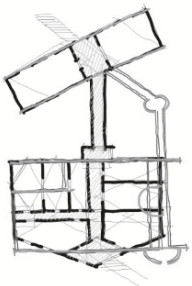
	Actor *A*	Actor *B*	Actor *C*
State *n*			
State *n+1*			
State *n+4*			
State *n+5*			

Figure 9. Syntagmatic parsing of synchronous case.

In Figure 9, state *n*+4, actor A commented unfavorably on the positioning of the main corridor that connected the two buildings, as it was sketched by

actor B in state *n*+1. He suggested that it should be transposed to the side of the main building in order to distinguish between two different types of pedestrian circulation. Actor B was actually inspired by this and transformed what he revealed to consider a disappointingly static design into a more flexible layout, which was analyzed favorably by actor A (state *n*+5). In a sense actors A and B were using each other's input in the same way annotations referred to a fixed design in the prints from a CAD representation.

Using the digital pen as a sketching tool was a logical consequence of such meetings, also in asynchronous situations: participants were encouraged to use their digital pens (including the information stored in them) also between meetings. This produced a number of elaborations of the ideas each actor had presented in the previous meeting, as well as reactions to ideas of other participants. Putting these together at the start of a meeting was a productive enrichment of approving the minutes of the previous meeting and setting up the meeting agenda.

Under these conditions the temptation to use the digital pen to sketch was irresistible. The results however were variable. Sketching with the digital pen is similar to sketching with ballpoint pen, Figure 10. The main ergonomic difference lies in the thickness of digital pen: the holding area has a circumference of approximately 60 mm compared to 30 mm for a pencil. Also the built-in LED camera has limited sensitivity: quick, short or light strokes are poorly captured, especially when the pen is held at an angle approaching 60° with respect to the paper. As a result the best digital pen sketches were fairly abstract and diagrammatic. As a sketching instrument for a single user the digital pen compares unfavorably to digital alternatives such as pressure-sensitive graphic tablets and related software in all respects but mobility and precision.

Figure 10. Digital pen sketches.

A significant limitation of sketching with the digital pen is that color and line weight are only visible in the digital version. As with mechanical digitizers the user is obliged to consult frequently the image in the computer. Unlike with mechanical digitizers this can be done only asynchronously, alternating sketching actions with synchronization and controlling the sketch in the computer. This lack of immediate visual feedback places restrictions to the use of the digital pen in artistic sketching.

A notable exception is another multi-user environment, education, where interaction between different actors (teachers and students) aims at elucidating and improving performance. In teaching activities relating to group design processes, morphological analysis or sketching, the explicitness of syntagmatic information also assists the analysis of the paradigmatic and mechanical dimensions. The ability to distinguish between strokes on the basis of the sequence in which they were made facilitates not only the recognition of accidental emergence but also the recognition of symbols composed by these strokes and the identification of drawing styles. For example, it helps analyze and structure the development of a freehand sketch, as in Figure 11 andFigure 12 (Cheng 2004; Cheng and Lane-Cumming 2003). A comparison of the two figures reveals that both sketchers used a similar syntagmatic strategy but different primitives in the basis of the drawing.

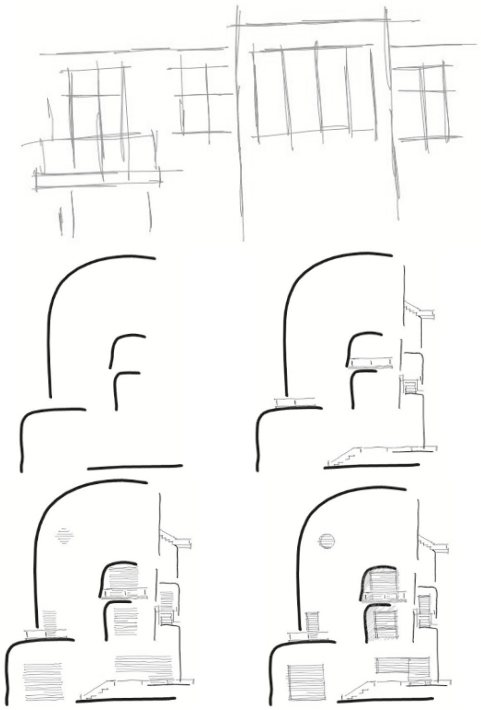


Figure 11. Drawing sequence in a freehand sketch.

Figure 12. Drawing sequence in a freehand sketch.

6. Conclusions

Our interest in digital sketching in multi-actor design environments derived from the need to identify and analyze individual input and relationships between different actors and corresponding aspects, actions and products. Even though we paid little attention to the paradigmatic dimension, the application of syntagmatic parsing returned unambiguous sequences of stokes with a clear authorship and interrelationships. This was generally sufficient for the reconstruction of group processes in a synchronous setting and led to a transparent interpretation of the content and intention of individual actions and decisions, as well as of different states of a design.

The approach and technology used proved well suited to the needs of annotating analogue versions of digital representations, as well as of abstract, diagrammatic sketching in early design. In other applications the results were variable. This was mainly due to the mechanical limitations of the current state of the digital pen. Its (cognitive) ergonomics suffer from the

pen size, the limitations of the built-in LED camera and the troublesome correlation of the advanced color images in the computer and the basic ballpoint drawings on paper. As usually in a digital environment, there is no single tool that is perfect for every task.

Our positive experiences with the technology used lie not in the replication of analogue means and procedures but in the integration of digital information processing in analogue situations. In the currently confusing interchangeability of digital and analogue versions of the same information, technologies that bridge the gap can be particularly useful. Mobility in information processing is essential to this, as it allows for more flexibility in the interaction between digital tools and analogue situations.

From a holistic viewpoint syntagmatic analysis is a prequel to the recognition of the paradigmatic structure of an image. The identification of meaningful symbols in a sketch is a prerequisite to any transformation into another representation, e.g. a measured drawing or a three-dimensional model. However, our analysis of the syntagmatic dimension reveals that such symbols may be formed by strokes that do not follow each other or may integrate strokes made by various actors (normally used by a single actor as a reference frame). Consequently the combination of discrete strokes and syntagmatic information may be insufficient for unambiguous recognition. The hypothesis that a model base of paradigmatic primitives forms the foundation of sketch recognition underlies the following stage of our research into digital sketching. The same applies to the recognition of symbols and forms produced by the combination of elliptical annotations with information already existing in the representation.

References

Achten, H, De Vries, B and Jessurun, J: 2000, DDDOOLZ, A virtual reality sketch tool for early design, *CAADRIA,* Singapore, pp. 451-460.

Achten, H and Jessurun, J: 2002, An agent framework for recognition of graphic units in drawings, *eCAADe*, Warsaw, pp. 246-253.

Cheng, NY-w: 2004, Stroke sequence in digital sketching, *eCAADe*, Copenhagen, pp. 387-393.

Cheng, NY-w: 2004, Teaching with digital sketching, *in* W Bennett and M Cabrinha (eds), *Design Communication Association - 11th Biannual Conference*, San Luis Obispo, pp. 61-67.

Cheng, NY-w and Lane-Cumming, S: 2003, Using mobile digital tools for learning about places, *CAADRIA*, Bangkok, pp. 145-156.

Do, EY-L: 2001, Graphics interpreter of design actions. The GIDA system of diagram sorting and analysis, *in* B de Vries, J van Leeuwen and H Achten (eds), *Computer Aided Architectural Design Futures*, Kluwer, Dordrecht, pp. 271-284.

Do, EY-L: 2001, VR sketchpad: Create instant 3D worlds by sketching on a transparent window, *in* B de Vries, J van Leeuwen and H Achten (eds), *Computer Aided Architectural Design Futures*, Kluwer, Dordrecht, pp. 161-172.

Do, EY-L: 2002, Drawing marks, acts, and reacts: Toward a computational sketching interface for architectural design, *AI EDAM* **16**(3): 149-171.

Gross, MD: 1995, Indexing visual databases of designs with diagrams, *in* A Koutamanis, H Timmermans and I Vermeulen (eds), *Visual Databases in Architecture. Recent Advances in Design and Decision Making*, Avebury, Aldershot, pp. 1-14.

Jozen, T, Wang, L and Sasada, T: 1999, Sketch VRML - 3D modeling of conception, *in* A Brown, M Knight and P Berridge (eds) *Architectural Computing: From Turing to 2000*, University of Liverpool, Liverpool, pp. 557-563.

Koutamanis, A: 1995, Recognition and retrieval in visual architectural databases, *in* A Koutamanis, H Timmermans and I Vermeulen (eds), *Visual Databases in Architecture. Recent Advances in Design and Decision Making*, Avebury, Aldershot, pp. 15-42.

Koutamanis, A: 2001, Prolegomena to the recognition of floor plan sketches: A typology of architectural and graphic primitives in freehand representations, *in* H Achten, B de Vries and J Hennessey (eds), *Design Research in The Netherlands 2000*, Faculteit Bouwkunde, TU Eindhoven, Eindhoven, pp. 93-103.

Sugishita, S, Kondo, K, Sato, H, Shimada, S and Kimura, F: 1995, Interactive freehand sketch interpreter for geometric modelling, *Proceedings of the Sixth International Conference on Human-Computer Interaction*, pp. 561-566.

Van Sommers, P: 1984, *Drawing and Cognition: Descriptive and Experimental Studies of Graphic Production Processes*, Cambridge University Press, Cambridge.

Woessner, U, Kieferle, J and Drosdol, J: 2004, Interaction methods for architecture in virtual environments, *eCAADe*, Copenhagen, pp. 66-73.

THE DESIGNOSAUR AND THE FURNITURE FACTORY

Simple Software for Fast Fabrication

YEONJOO OH, GABERIAL JOHNSON, MARK D GROSS AND ELLEN YI-LUEN DO
Carnegie Mellon University, USA

Abstract. We describe two domain oriented design tools that help novice designers design three-dimensional models that they can build using rapid manufacturing equipment. By embedding domain and manufacturing knowledge in the software and providing a sketching interface, novice designers can acquire and practice skills in modeling and manufacturing, without first having to master complicated CAD tools.

1. From Sketching to Fabrication: Simple Design Tools for Making Things

1.1. INTRODUCTION

We want to make it easy for ordinary people, especially children, to design and manufacture three-dimensional models using planar components as an entrée to learning to design. We believe that the experience of designing and making things is a powerful vehicle for learning. For many people, designing and making something can be rewarding and engaging, and, we think, can motivate more general learning in science, technology, engineering, and mathematics. Further, we think that in its own right, designing is an intellectual capacity that, once acquired in one domain, can be widely applied. Sadly, many people view design as an innate talent for creativity that they lack. Our project to build lightweight software to engage young people and naïve users in designing and manufacturing simple 3-D models aims to open the door to design and the rich universe of learning that design affords.

Three-dimensional physical models are powerful devices that help people see and understand designs. One can hold a physical model in the hand, take it apart, and reassemble it, perhaps in different ways. This ability to interact physically with a model and its parts is important, we think, for thinking about a design; and the experience of designing with 3-D models teaches

J.S. Gero (ed.), Design Computing and Cognition '06, 123–140.
© 2006 *Springer. Printed in the Netherlands.*

spatial skills that designers cannot easily acquire through other means such as drawing or computer graphics modeling.

Making models in the traditional way demands considerable manual skill and dexterity, for example cutting wood parts with a razor knife. The advent and adoption of rapid prototyping and manufacturing (RPM) machinery has made it possible for ordinary designers, students, and even children, to produce physical artifacts using computational means. Although most RPM hardware is as simple to use as a printer, the software tools that designers use to produce representations for output require a great deal of expertise. To produce a 3-D model designers must create a computer graphic representation. Typically designers do this using powerful general-purpose CAD modeling tools that impose a significant learning curve. Requiring of professional designers this degree of sophistication and expertise may be acceptable; however, the tools bar entry to casual and novice users.

Our goal in this work is on one hand to explore and populate the space of computationally enhanced craft activities for teaching and learning design; and on the other hand, to develop interaction techniques appropriate to the domain and the intended users. We aim to exploit characteristics of specific design domains to build relatively simple special-purpose modeling tools that novice users can easily and quickly learn. The computational design environment can help a designer by building in some requirements of the manufacturing method (e.g., the method of joining parts and sizing of joint features) as well as circumscribe the design domain through the methods and options the tools present. Our method is to explore this territory through constructing working prototypes that illustrate ideas and possibilities.

1.2. FURNITURE AND DINOSAUR MODELS

We describe two projects that explore this territory of easy-to-use domain-oriented tools that help naïve users design three-dimensional models. Using the Furniture Factory users sketch, model, and produce a class of furniture models. The Designosaur enables its users to make models of dinosaurs, both real and imaginary. Both projects enable users to begin by sketching a design and both employ a laser cutter as the output medium, to cut model materials from flat material such as wood or plastic sheets. Both use embedded knowledge about the design domain and manufacturing process to implicitly help users make design decisions or add detail to the model.

Figure 1 shows a store bought model furniture kit. It is packaged as flat panels with pre-cut parts that can be punched out and assembled into furniture models. A simple joining system (note small holes and slots in the laid-out parts in Figure 1) keeps the assembled parts together without glue or additional hardware. If permanence is desired, the parts can be glued and the model can be painted. Some furniture models are cleverly designed. For example, the cabinet doors in the piece shown below are hinged to open and

close: The small tabs on the top and bottom of the two door parts (bottom right) fit into holes in the two lower horizontal shelves (bottom middle).

Similar in manufacture to the miniature furniture kits are off-the-shelf dinosaur kits. These consist an assortment of wooden "bones" that can be put together as though it were a puzzle, Figure 2. It is satisfying to bring form to a three-dimensional creature by putting together bits of wood. Like the furniture kits, all the parts are planar; they join orthogonally using a paired notching scheme to make a three-dimensional structure.

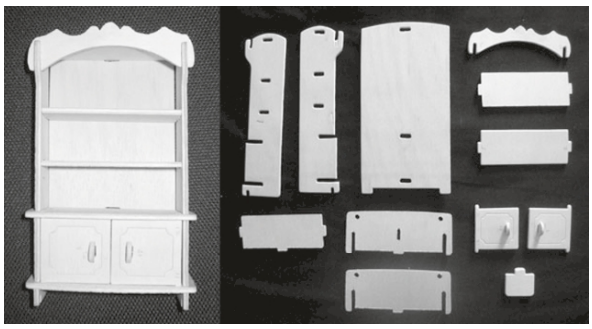


Figure 1. A furniture model kit packaged as flat panels of punch-out parts with notches and tabs for joints.

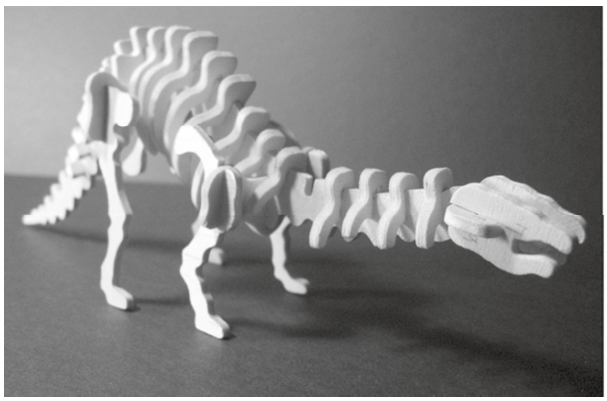


Figure 2. Dinosaur model made from a kit.

As cute (or terrifying) as these models are, a drawback to both these kits (and other off-the-shelf wood models) is that you are limited to assembling a figure that someone else has designed. When you are finished, you have a dinosaur skeleton that looks good sitting on a bookshelf, or a set of furniture that you can put in your dollhouse.

To be sure there is satisfaction in successfully assembling a model; and some understanding to be gained in seeing how the flat parts go together to make a 3-D form. Yet after a few successful attempts, one begins to think

about how a model could be changed: What if you could create something that might not have ever existed, such as a three-headed raptor with wings? What if you could design your own cabinet or table, instead of accepting the stuffy conventional models that come in the box? In short, with a store-bought model you are limited to the manual labor of assembling the parts leaving no room for you to participate as a designer.

Nothing prevents an enterprising young designer from going to the local art-and-craft supply store and buying a sheet of basswood, and cutting out a model of his or her own design. However, the step from purchase-and-assemble to building-from-raw-materials is large: The designer must decide and draw the parts, determine how they will join, then carefully cut them from the sheet. Tolerance errors have severe consequences; a small mismatch between notch size or position and material thickness means the model won't assemble correctly. The process also requires considerable dexterity with a razor knife, a skill that takes some time to acquire. All these are, of course, part of designing and manufacturing a wood model in the traditional way. It may be argued that eliminating these components "dumbs down" the process and eliminates learning opportunities. On the contrary, we believe that at least for beginning designers it is important to put design decision making first, and by providing scaffolding for decision making and manufacturing, we can draw them into the game.

1.3. RELATED WORK

Our project relates to research in several areas: learning through design, feature modeling and design assembly, and sketch-based 3-D graphics.

Activity learning theory stresses the importance of doing things to learn (Blust and Bates 2004). People engage more in learning when they act than when information is presented for their passive consumption. Problem-based learning is a widely cited example, in which teams of students work together to address real-world problems. The studio model of learning in architecture and product design depends on hands-on experience for students to acquire knowledge and skills with the guidance of a design instructor. Others argue that design is a powerful vehicle for learning mathematics, using Escher's drawings or quilt pattern design as domains (Shaffer 1997; Lamberty and Kolodner 2002).

We follow in the "constructionist" tradition of mathematician and educator Seymour Papert and his students and colleagues at MIT. In developing the Logo programming language for children Papert argued that programs children write function as mathematical 'objects to think with' and that their concreteness affords a powerful kind of learning (Papert 1980). Later work of Papert's former students and colleagues (e.g. Mitchel Resnick's group at MIT) has explored and extended this idea, moving beyond the inner world of software into the hybrid world of physically embodied computation (Resnick et al. 1998). Michael Eisenberg and his

craft technology group, our collaborators at the University of Colorado, have explored a variety of computer assisted physical construction and craft activities, beginning with computational tools for designing and making polyhedron models out of paper (Eisenberg and Eisenberg 1997), and more recently mechanical automata toys (Blauvelt et al. 2000) and electronically embedded quilts and clothing (Buechley et al. 2005). We have seen an explosion of interest in professional architecture circles and among architectural educators in "digital fabrication", the use of rapid prototyping and manufacturing to produce buildings and models (Kolarevic 2004, Sass 2005, Iwamoto 2004). Neil Gershenfeld (Gershenfeld 2005) discusses the more general implications of these technologies.

The second area of related work is feature modeling and design assembly. The two projects described here are reminiscent of the feature-based modeling and design for assembly or design for manufacturability research prevalent in the late 1980s and 1990s in the AI and Design community (Dixon et al. 1987; Shah 1991; Shah and Mantyla 1996). Jung and Nousch (2000) describe a system for designing and fabrication of furniture using computer graphics modeling. Their BEAVER program allows users to construct closets of various types and sizes in a 3-D computer graphics environment using parametric modeling. It details all necessary parts and hardware and it creates assembly instructions with text and step-by-step drawings. Also in the domain of furniture parts assembly, Agrawala et al. (Agrawala et al. 2003) studied people following printed assembly instructions and diagrams and based on this study described principles for better diagrams. Their system automatically generates appropriate assembly diagrams according to these principles.

The third area of related work deals with creating 3-D models from 2-D sketches. Recently software has been developed that can construct a 3-D object from a 2D sketched drawing. For example, Teddy (Igarashi et al. 1997) models 3-D curved surfaces (such as a Teddy bear) from 2D freeform strokes. It creates a closed polygon from a stroke and finds a spine in center of the polygon. Next, it creates a 2D triangular mesh between the spine and the polygon perimeter, and then raises each spine vertex and smoothes the mesh. Digital Clay (Schweikardt and Gross 2000) converts 2D representation of edges into a 3-D Model using the Huffman-Clowes scheme to label each edge in a drawing as a concave or convex junction of faces. 3-D Journal (Lipson and Shpitalni 2002) constructs a 3-D model from 2D strokes and allows a user to change viewpoint while sketching. First it calculates the angles of all segments in a sketch and identifies a prevailing axis system. It then assigns all lines with the identified axis system, tolerating inaccurate vertex positions, inaccurate connections between segments and sketches with missing lines. The axially-aligned planes approach of (Varley et al. 2005) inflates a wireframe drawing to a 2-1/2 D

model by labeling lines as convex and concave, determining pairs of parallel lines, and calculating a Z-coordinate value for each vertex. In Furniture Factory, we assume a prevailing 3-D axis system and match lines in the sketch to axial lines—similar to the techniques of Lipson and Varley.

2. Furniture Factory

The Furniture Factory program is a proof-of-concept working prototype built to explore how to help designers construct physical prototypes using rapid prototyping and manufacturing machines. It provides a sketch-based design interface that a designer can use to draw furniture in 3-D. The design is then displayed in an isometric viewing window where the designer can view it and edit it. The program then decomposes the 3-D model into flat panels that are displayed in the parts window. Furniture Factory adds joints where one panel connects to another according to connection conditions. These added joints enable designers to construct the physical model easily and quickly. The program then generates HPGL code to cut the furniture parts on a laser cutter. Designers construct their furniture by assembling the cut parts.

Furniture Factory has four components that share a data structure: Sketch Interface, Geometry Analyzer, 3-D Representation, and Joint Creator. Figure 3 shows the information flow among these components After a designer sketches the isometric view of furniture in the Sketch Interface, Geometry Analyzer analyzes the sketch and computes faces, edges and 3-D coordinates. These computed faces, edges, and their 3-D coordinates are stored in the data structure. 3-D Representation creates a 3-D model using information extracted from the designer's isometric sketch and displays it. The Geometry Analyzer also decomposes the 3-D model into 2D geometries and analyzes the connection conditions. These decomposed geometries are also stored in Data Structure. Joint Creator adds necessary joints according to the identified connection conditions.

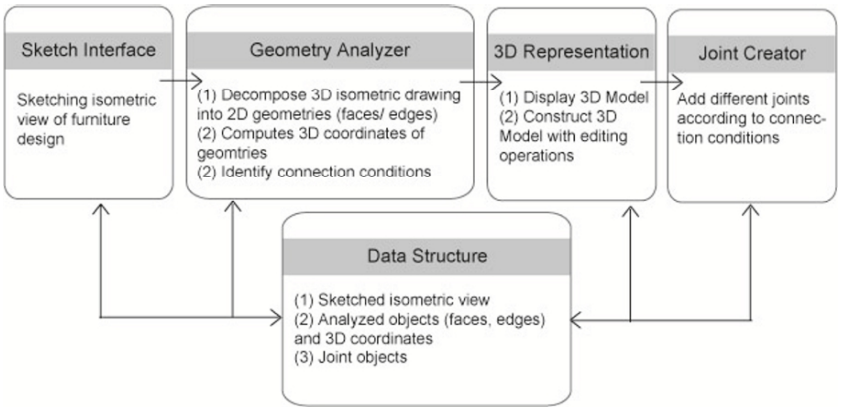


Figure 3. Components of Furniture Factory.

The Furniture Factory user interface employs three windows. The *Sketch* window displays freehand sketches the designer draws with a tablet and stylus, Figure 5(a). The *3-D* window has two functions: to enable the designer to view the design and to edit its parts. Figure 5(b) shows a 3-D view of the bookshelf generated from the sketch. These parts can also be moved and rotated. The *Parts* window displays the parts and their joints, Figure 9(a). The rest of this section describes these components and their roles in Furniture Factory.

2.1. SKETCHING INPUT

In Furniture Factory's freehand sketching environment designers use a stylus and tablet to make freehand isometric sketches to create 3-D models, Figure 5(a). Our simple sketch interface recognizes three types of lines: horizontal, vertical, and diagonal. Following isometric conventions each type of line is assigned to one of the Cartesian coordinate axes: horizontal line to the Y-axis, vertical line to the Z-axis, and diagonal line to the X-axis. Figure 4 illustrates the relationship between 2D directions of lines and the 3-D coordinate axes.

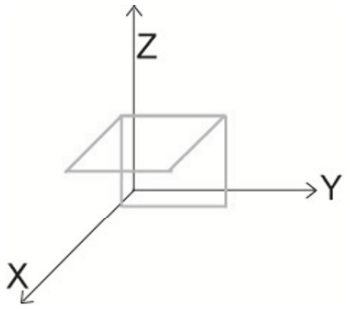


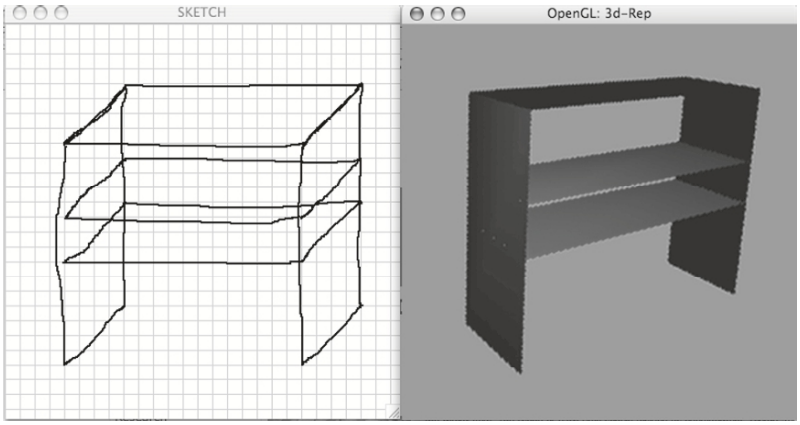
Figure 4. Relationship between three types of lines and 3-D Cartesian coordinates.

2.2. 3-D REPRESENTATION

Furniture Factory provides the designer with a 3-D display window in which the sketched model converted to 3-D data objects is displayed using an OpenGL viewer, Figure 5(b). The 3-D representation displays a 3-D model of a sketched artifact. The designer can rotate the model to visually evaluate the furniture design, Figure 5(b).

2.3. DATA STRUCTURE

Furniture Factory stores three kinds of data objects: faces, edges, and joints. Figure 6 shows the objects, their attributes, and their relationships. As the designer makes the sketch drawing, the Geometry Analyzer generates descriptions of the 3-D edges and faces and links them in the data structure to reflect the configuration of the furniture object.



(a) (b)

Figure 5. Sketching input and 3-D Representation: (a) Sketched isometric model of furniture; (b) 3-D Representation window displays 3-D representation.

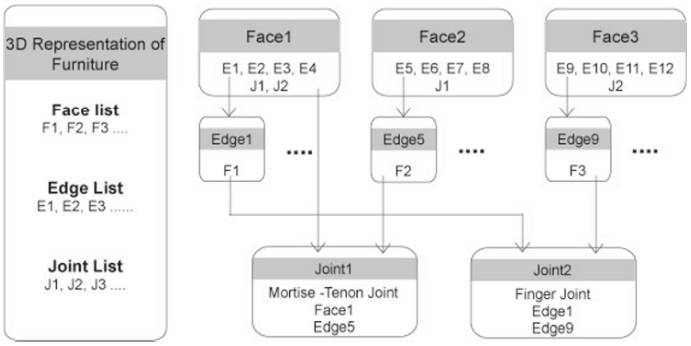
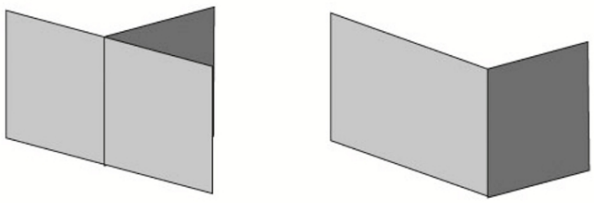


Figure 6. Objects and relationships: Each 3-D representation stores interconnected faces, edges, and joints.

2.4. GEOMETRY ANALYZER

Geometry Analyzer performs three functions: analyze the designer's isometric sketch, decompose it into basic elements (faces and edges) and determine how the faces and edges connect. First, Geometry Analyzer takes the isometric wire frame drawing from the Sketch Interface. Considering the wire frame drawing as a graph in which lines are edges and nodes are vertices of the 2D drawing, the Geometry Analyzer identifies simple cycles that contain no edges as faces and assigns them 3-D coordinates, using the isometric axis assignment heuristic mentioned above. Second, the Geometry Analyzer transfers each part elements into 2D coordinates to cut on a laser cutter. Third, the Geometry Analyzer identifies where and how these faces and edges connect to each other. The current Geometry Analyzer identifies

two kinds of connection conditions: face – edge and edge – edge. Figure 7 illustrates two connection conditions.



(a) (b)

Figure 7. Two connection conditions: (a) face::edge: one face abuts an edge of another face (b) edge::edge: when one edge forms a corner with an edge of another face.

2.5. JOINT CREATOR

Joint Creator adds joints into the furniture parts according to the identified connection conditions. Currently the Joint Creator supports two types of joints: (1) mortise-and-tenon joint and (2) finger joint. When one face connects to one edge of another face, the Joint Creator adds a mortise-and-tenon joint. When two faces connect at their edges in a corner, the program adds finger joints. Figures 7 and 8 illustrate connection conditions and joint types.

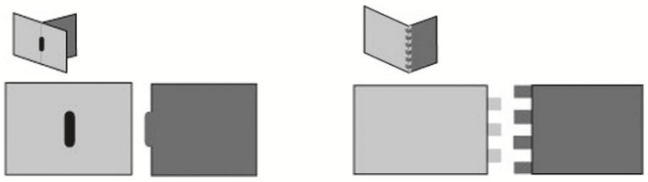


Figure 8. Connection conditions and joint types: Where a face and an edge connect, Furniture Factory creates a mortise and tenon joint. Where two edges connect in a corner the program creates a finger joint.

2.6. CUT AND ASSEMBLY

The Furniture Factory then displays the 2-D parts and their newly created joints in the Parts window. The Furniture Factory labels the faces and edges to help the designer assemble their furniture. For example, it labels two faces of a bookshelf "Bottom" and "Back". These two faces meet at a joint the system names "e". The system labels the two meeting edges – "e1" (Bottom) and "e2" (Back). Users can understand how to assemble parts

easily using these labels. Then Furniture Factory generates HPGL code to cut the furniture parts on a laser cutter ready to assemble. Figure 9 shows the drawings of parts, the cut panels and assembled furniture.

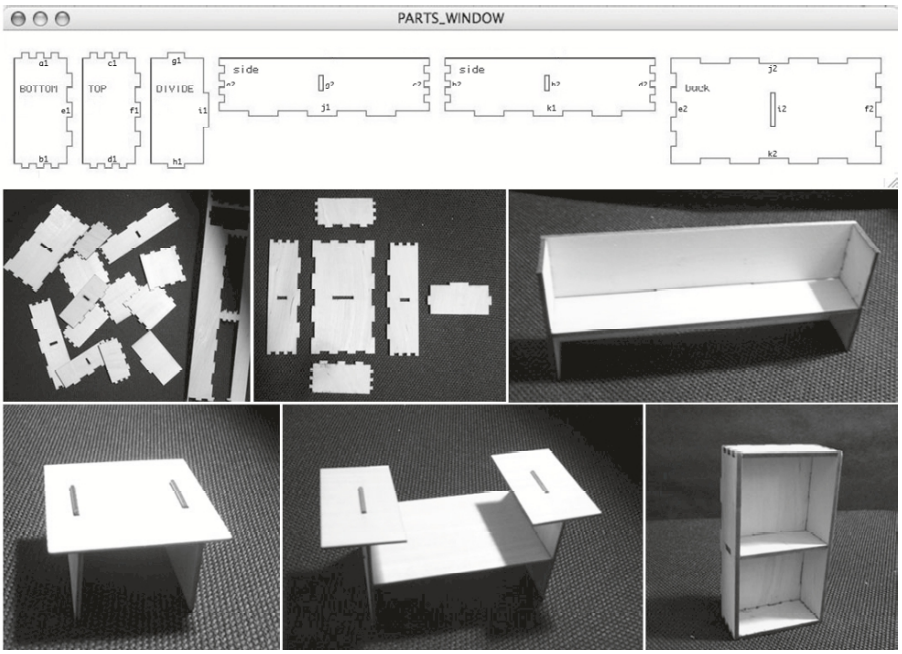


Figure 9. Drawing and parts on a laser cutter and assembled furniture: (a) Cut drawing is composed by labelled parts and joints; (b) the user assembles these parts and constructs physical furniture model.

3. The Designosaur

3.1. SKETCHING BONES

The Designosaur is a simple modeling program that empowers people to create their own models of dinosaurs or dinosaur-like skeletons. A Designosaur creature is assembled out of pieces of thin wood, plastic, foam, or other material. The pieces fasten together using a notching system with identical notches; pieces slide together in 'X' fashion such that their planes are orthogonal. Figure 10 shows a dinosaur leg bone sketched freehand in the Designosaur modeling program.

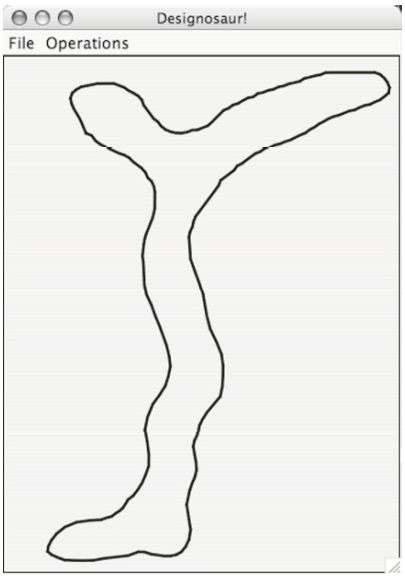
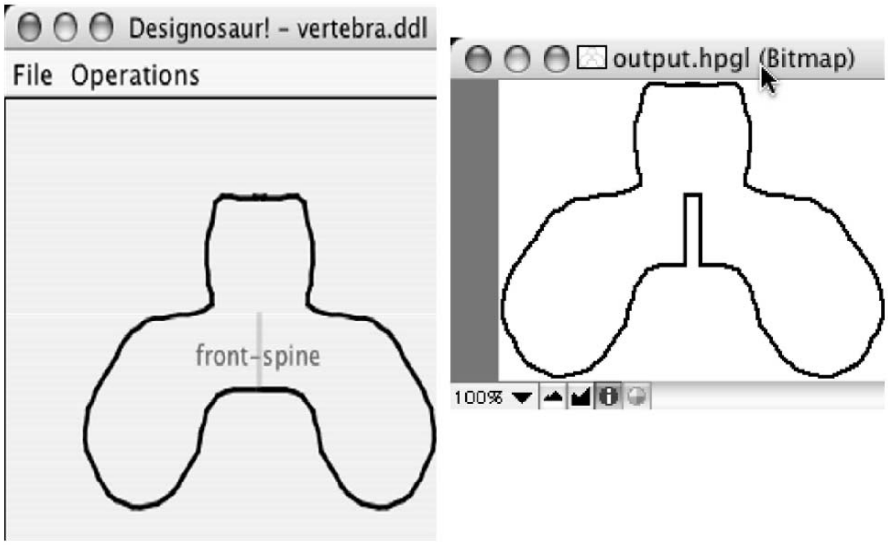


Figure 10. A dinosaur hind leg bone sketched in the Designosaur modeling program.

3.2. BONES ARE CONNECTED BY NOTCHES

In Designosaur, designers draw individual bones and indicate where other bones attach by specifying notch points. A notch is a joint that connects exactly two bones. Notch parameters (width and depth) are provided later on, when the designer is ready to produce a physical model on the laser cutter. For this reason, notches are displayed only as thin markers. In figure 11(a), the designer has drawn a vertebra bone, and inserted a notch along the bottom surface. The notch symbol points inward, indicating the direction of the cut. Figure 11(b) shows the notch detailed with width and depth appropriate for the chosen material, ready to print in an HPGL file.

If a notch knows which two bones it joins together, we can use this information to assemble a list of bones for printing, and also to construct a 3-D graphics model of the skeleton. The designer can identify the mating parts (by selecting from a menu of already-known parts) when notching a bone, or later in the process if the mating bone part has not yet been designed. The label appears as soon as the mating information has been added. Figure 14(b) shows a sketched dinosaur spine, with five notches. One notch is labeled “head connector” and the other four are labeled “vertebra”. The label next to each notch indicates the name of the mating part that the notch will join in the assembled model. Similarly, Figure 14(c) shows a symmetric vertebra with a notch labeled to show that the vertebra connects to the front spine bone.



(a) (b)

Figure 11. (a) Using the Designosaur's bone editor to create a vertebra. A thin line marks the location of the notch; (b) the same vertebra rendered in HPGL for cutting. The notch has been assigned a width and depth according to the chosen material.

3.3. SYMMETRY / MIRROR MODE

Many bones are symmetric, so the Designosaur supports symmetry in bone design by providing a 'mirror mode'. When the designer enters mirror mode, a vertical line is positioned down the centre of the drawing area. All the designer's strokes are duplicated on the opposite side of this mirror line. This helps with making bones such as ribs or vertebrae. The hip bone in Figure 12 was sketched in mirror mode.

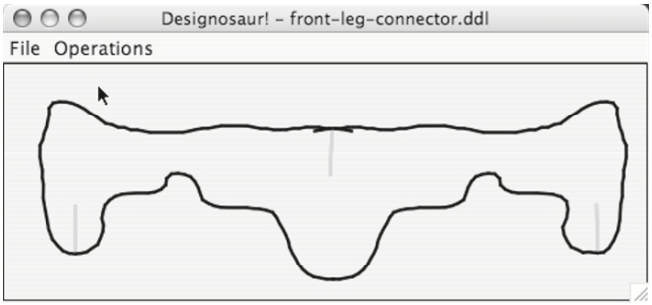


Figure 12. A front leg connector (hip), drawn using the mirror mode.

3.4. CONNECTION LINKS

The connection information that the designer provides by labeling notches with their mating parts enables the Designosaur program to construct a graph data structure that represents the connectivity of the bones in the model, Figure 13.

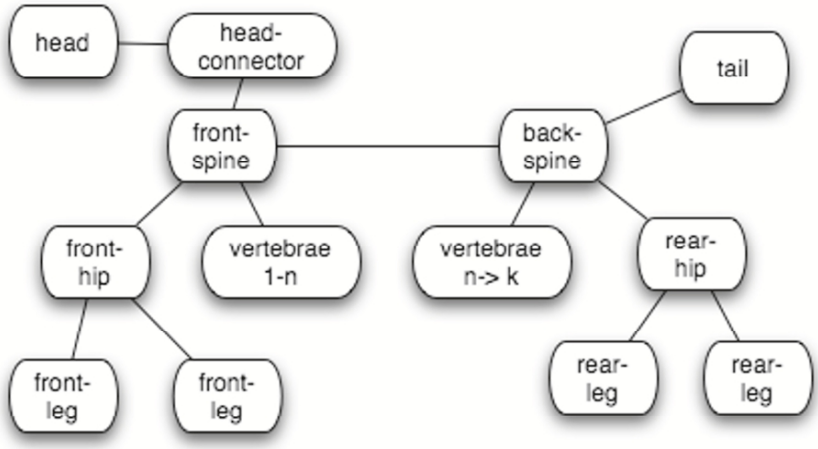
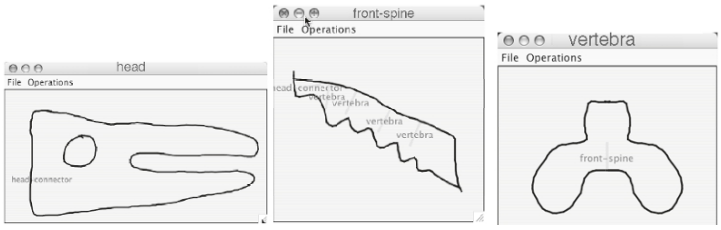


Figure 13. Structure of a dinosaur model.

3.5. NOTCH REGIONS

A dinosaur spine generally has a number of attached vertebrae. In order to help evenly distribute notch points for attaching these bones to the spine, the designer can specify 'notch regions'. The designer indicates how many notches to place along a given path, and the Designosaur automatically distributes the attachment points with even spacing. Figure 14(b) shows a spine to which the designer has applied four notches using a parameterised notch region.



(a) (b) (c)

Figure 14. Three bones and their relationships (shown by labelled notches), (a) head, with a notch for a head connector (not shown); (b) front-spine with four vertebrae notches and a notch for the head connector; (c) symmetric vertebra with notch for spine.

3.6. SYSTEM ARCHITECTURE

The Designosaur interprets sketch input, storing each stroke as custom point objects in a bi-directional linked list. The linked list structure provides a convenient method for traversing each stroke forwards or backwards. The custom point objects store location as well as notch data and they provide methods to get information about the stroke at that point, such as curvature or distance along the line to another point.

When the designer positions a notch, the point closest to the indicated location is flagged as having a notch that extends inward from the bone edge at that point. A notch is always normal to the edge. As there are two possible orthogonal directions to the stroke at any point, the user disambiguates direction by specifying a left- or right-handed normal vector. If the designer later edits the surface of the bone and moves the notch point, the left- or right-handedness remains constant, and the notch moves along with the stroke in an intuitive manner.

3.7. MANUFACTURING

After the bones and their connections have been designed, it is time to print the dinosaur using a laser cutter, Figure 15. The width of notches depends on the material of manufacture. For example, we have experimentally determined that 4 mm-thick foam core media (a rigid art material consisting of a sandwich of foam plastic between two paper faces) requires a notch that is 18 pixels wide on-screen. When printing bones, the Designosaur asks how thick the material is and uses that number to "notchify" (add notches to) the shape. The completed form, Figure 16.

Figure 15. Physical dinosaur parts: The foam core sheet from which the pieces are cut sitting on top of the laser cutter. Below, unassembled dinosaur bones.

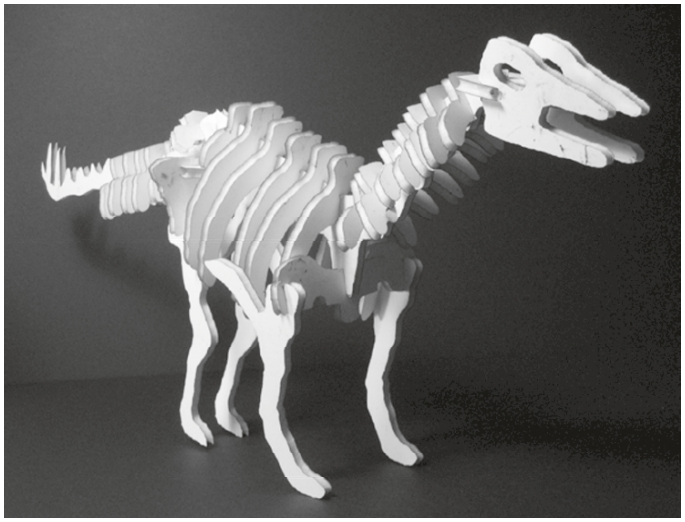


Figure 16. Made with the Designosaur: The authors created this dinosaur using the Designosaur system and a laser cutter. Compare with store-bought model, Figure 2.

4. Discussion and Future Work

4.1. DISCUSSION

We do not intend the Designosaur or the Furniture Factory as general-purpose design tools. Both are limited to constrained domains (furniture and dinosaur models) and a constrained manufacturing universe: assembling pieces cut from flat material using a small number of jointing techniques. For beginning designers, the limited domain and the constrained manufacturing method offer a design space that is at the same time rich in possibility yet not overwhelmingly large and complex. Both, of course, are specific instances of a more general design system in which 2-D parts are assembled using mating joints, which suggests a future effort to build a meta-modeler (a language for describing design systems) in which one can specify the Furniture Factory, the Designosaur, or other instances of flat single-material component assembly systems.

There is a delicate balance between doing the work for the designer, and requiring the designer to do the work. On one hand, if our goal were merely to enable people to produce 3-D models, we could provide a catalog of parameterized designs. A user would select a desk or a dinosaur, choose parameters, such as the overall dimensions, and then print (cut) the model. Although this satisfies the requirement of producing 3-D models and in our limited universe it might be sufficient to cover the design space, this would deny users the experience of designing.

On the other hand, a designer could simply draw model parts directly, using a simple 2-D drawing program. As the models are all cut from flat material and the files sent to the laser cutter are 2-D drawing files, this wouldn't be too difficult, at least in principle. However, the designer would need to add jointing details to the parts. Although conceptually simple, it is a tedious task to get the sizes and positions of the joints exactly correct. The Furniture Factory also provides the user with a 3-D view of the assembled design unavailable in a 2-D drawing program.

Our projects contribute to the field of personal fabrication by providing sketching interfaces and intelligent design support, and we hope to gain insight into creative processes and learning through developing personal fabrication technology. Striking the balance between preserving simplicity while supporting creativity is a challenging goal for this research. We described here two example systems that can be considered instances of a more general design system. This general design system comprises three parts: (1) sketching input and recognition, (2) inference, interpretation and design decision support, (3) transformation and preparation for machine fabrication of parts. Our projects are still proof-of-concept working prototypes that we hope to develop into more robust systems that we can evaluate with end users; at this point, though, our assessment of these prototypes has been solely in the nature of formative evaluation leading to incremental improvements of the human-computer interface.

4.2. EXTENDING THE FURNITURE FACTORY AND DESIGNOSAUR

We plan to add additional jointing conditions to Furniture Factory, which will allow designers to construct more complex objects than those allowed by the current finger and mortise-and-tenon joints. A richer vocabulary of joints could also allow the designer to select among alternative jointing methods: in the current version the program determines the jointing scheme, although the designer can override the system. We also plan to add an error-checking and critiquing component, building on our previous design critiquing work (Oh et al. 2004). For example, if the designer sketches a table with different length legs, the Furniture Factory's critics could warn the designer that the table will be unstable. Beyond simple error checking, the Furniture Factory's critiquing component could retrieve interesting or relevant cases from a library of stored models.

The current Designosaur provides the ability to draw simple bones, indicate where they fit together, then generate HPGL code needed to produce the physical output. As with the Furniture Factory, simple critiquing both about the domain (the dinosaur won't stand up: move its center of gravity) and about fabrication would be a valuable addition. Beyond small improvements to the interface that we expect to identify when we test the

system with users beyond graduate students in our laboratory, we plan to add a 3-D modeling graphics view, similar to that offered in Furniture Factory. Looking further ahead, we envision a sketch-based system that allows users to draw the outside shape of a creature and let the system infer internal skeletal structure. For example, a drawing of a dinosaur that has a long neck, a fat midsection, a tail, and two nubs for legs is enough for a shape recognition system to determine the general features of the creature and make some intelligent guesses about what the designer wants. The size of the ribs, for example, will conform to the curvature of the belly, so that the ribs in the middle are longer and wider than those towards the front and hind legs. Individual bones will also be malleable; the designer will be able to grab the tail and yank it away from the body in order to make it longer, and the Designosaur will add vertebrae to the spine as necessary. More speculatively, it would be possible for the designer to add mechanical joints to the dinosaur, test the kinematics in an animation on the screen, and then print out the parts and assemble a moveable model. One could even imagine adding electronics and servomotors to the model to make the dinosaur walk on its own. But that is another story …

Acknowledgements

We thank the Institute for Complex Engineered Systems (ICES) at Carnegie Mellon University for support. This work is supported in part by the Pennsylvania Infrastructure Technology Alliance (PITA), a partnership of Carnegie Mellon, Lehigh University, and the Commonwealth of Pennsylvania's Department of Community and Economic Development (DECD). This research was supported in part by the National Science Foundation under Grant ITR-0326054. The views and findings contained in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Arawala, M, Phan, D, Heiser, J, Haymaker, J, Klingner, K, Hanrahan, P and Tversky, B: 2003, Designing effective step-by-step assembly instructions, *SIGGRAPH 2003*, pp. 828-837.

Blauvelt, G, Wrensch, T and Eisenberg, M: 2000, Integrating craft materials and computation, *Knowledge Based Systems* **13**:471-478.

Blust, RP and Bates, JB: 2004, Activity based learning - Wagons R Us - A lean manufacturing simulation, *ASEE Annual Conference and Exposition: Engineering Education Reaches New Heights*, Salt Lake City.

Buechley, L, Elumeze, N, Dodson, C and Eisenberg, M: 2005, Quilt snaps: A fabric based computational construction kit, *IEEE International Workshop on Wireless and Mobile Technologies in Education*, Available Online:

http://doi.ieeecomputersociety.org/10.1109/WMTE.2005.55

Dixon, JR, Cunningham, JJ and Simmons, MM: 1987, Research in designing with features, *in* H Yoshikawa and D Gossard (eds), *Proceedings IFIP TC 5/WG 5.2 Workshop on Intelligent CAD*, New York, pp. 137-148.

Eisenberg, M and Eisenberg, AN: 1997, Orihedra: Mathematical sculptures in paper, *Journal of Computers in Mathematics and Science Teaching* **16**: 477-511.

Gershenfeld, N: 2005, FAB: *The Coming Revolution on Your Desktop--From Personal Computers to Personal Fabrication,* Basic Books.

Igarashi, T, Matsuoka, S and Tanaka, H: 1999, Teddy: A sketching interface for 3D freeform design, *ACM SIGGRAPH'99*, Los Angels, pp. 409-416 .

Iwamoto, L: 2004, Embodied fabrication: Computer-aided spacemaking, *in* P Beesley, NY-W Cheng and RS Williamson (eds), *Association for Computer Aided Design in Architecture: Fabrication: Examining the Digital Practice of Architecture ACADIA*, Toronto, pp. 269-281.

Kolarevic, B: 2004, *Architecture in the Digital Age: Design and Manufacturing,* Spon, London.

Lamberty, KK and Kolodner, JL: 2002, Exploring digital quilt design using manipulatives as a math learning tool*, International Conference of the Learning Sciences*, Seattle, WA, pp. 552-553.

Lipson, H and Shpitalni, M: 2002, Correlation-based reconstruction of a 3D object from a single freehand sketch, *AAAI Spring Symposium on Sketch Understanding*, Washington DC, pp. 99-104.

Oh, Y, Do, EY-L and Mark D Gross: 2004, Intelligent critiquing of design sketches, *in* JL Randall Davis, T Stahovich, R Miller and E Saund (eds), *Making Pen-Based Interaction Intelligent and Natural*, AAAI Press, Arlington, Virginia, pp. 127 - 133.

Papert, S: 1980, *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, New York.

Resnick, M, Martin, F, Berg, R, Borovoy, R, Colella, V, Kramer, K and Silverman, B: 1998, Digital manipulatives: New toys to think with, *SIGCHI Conference on Human Factors in Computing Systems*, Los Angeles, CA, United States, pp. 281-287.

Sass, L: 2005, Wood frame grammar: CAD scripting a wood frame house, *CAAD Futures*, Vienna, Austria, Available Online: http://ddf.mit.edu/papers/06_lsass_caad_future_2005.pdf

Schweikardt, E and Gross, M: 1988, Digital clay: Deriving digital models from freehand sketches, *ACADIA,* Quebec City, Canada, pp. 202-211.

Shaffer, DW: 1997, Learning mathematics through design: The anatomy of Escher's wWorld, *Journal of Mathematical Behavior* **16**: 95-112.

Shah, JJ: 1991, Conceptual development of form features and feature modelers, *Research in Engineering Design* **2**(2): 93-108.

Shah, JJ and Mantyla, M: 1996, *Parametric and Feature-Based CAD/CAM*, John Wiley, New York.

Varley, PAC, Martin, RR and Suzuki, H: 2005, Progress in detection of axis-aligned planes to aid in interpreting line drawings of engineering objects, *in* T Igarashi and JA Jorge (eds), *Eurographics Workshop on Sketch-Based Interfaces and Modeling,* Dublin, Ireland, pp. 99-108.

INDUSTRIAL MECHANICAL DESIGN: THE IDS CASE STUDY

STEFANIA BANDINI AND FABIO SARTORI
University of Milan-Bicocca, Italy

Abstract. This paper presents IDS, a knowledge based system to support experts in the design and manufacturing of complex mechanical objects. Although many sophisticated integrated environments have been developed and are currently adopted by enterprises in the design of mechanical parts and components, they can be extended to further support design activities. Artificial Intelligence techniques are particularly suitable to provide such tools with knowledge based facilities derived from skilled designer's expertise. IDS (Intelligent Design System) is a set of integrated AI-based facilities interacting the CATIA CAD software environment to support the design of dies for car body manufacturing.

1. Introduction

Intelligent Design System (IDS) is a set of integrated AI-based facilities to support design activities devoted to the production of complex industrial mechanical objects. Although many sophisticated CAD environments have been developed and are currently adopted by enterprises in the design of mechanical parts and components, such kinds of tools can be extended in order to improve the performance of their users. In this paper, the case of application of IDS and its integration within the CATIA (see e.g. Nishimoto et al. 1991) software environment for supporting design activities in a real industrial case will be presented. The enterprise which funded and that is currently using the IDS solution is *Fontana Pietro S.p.A.*[1] the Italian enterprise leader in dies manufacturing and construction of elite car bodies.
In the next Section a brief description of a die will be given (in order to provide the basic knowledge to be used to illustrate the main features of The IDS system), in order to introduce the conceptual and computational framework that inspires the adopted software solution.

Section 3 will focus on some characteristics of the knowledge model that has been created in order to capture functional, procedural and experiential

[1] http://www.fontanapietrospa.com

J.S. Gero (ed.), Design Computing and Cognition '06, 141–160.
© 2006 *Springer. Printed in the Netherlands.*

knowledge characterizing the application domain. Section 4 is devoted to briefly describe the implementation features of the system, while the last two sections will present some considerations about the main practical obtained results, and concluding remarks.

2. A Challenging Domain for Intelligent Design Support Systems

The development of IDS (Intelligent Design System) as a solution for integrating experiential knowledge shared by professional designers has been stimulated and supported by a company that is the Italian leader in dies manufacturing for elite automotive industries. The pressure of time-to-market requirements and globalization competitiveness pushed the top management of this company to investigate how knowledge coming from research could improve the performance of the Design Unit. It is a crucial unit whose performance heavily influences both costs and related production activities (e.g., simulation, manufacturing). The investigation on how to improve the Design Unit production activity led to study the possibility of the application of methods and techniques coming from Artificial Intelligence, in order to develop knowledge-based facilities. The design and development of the IDS solution is the result of the intensive collaboration between the Artificial Intelligence Lab (University of Milano-Bicocca), and the deep involvement of the top management and the design professionals of *Fontana Pietro* company in all the project steps.

2.1. KNOWLEDGE FOR DIE DESIGN: AN OVERVIEW

A die is a very complex mechanical product, composed of a huge number of parts with different functions that must be put together into a unique and homogeneous steel fusion. Each die is the result of a complex design and manufacturing process involving many professionals. Moreover, it is important to notice that more dies are necessary to produce a car body, each one having precise and distinct roles within the whole process.

The car body is the result of a multi–step process in which a thin sheet metal is passed through different kinds of presses, each one equipped with a specific die. Four main kinds of dies can be identified:

- Forming die: it provides the sheet metal with the final morphology of the car body die;
- Cutting die: it cuts away the unnecessary parts of the sheet metal;
- Boring die: it makes holes in the sheet metal, in order to make it lighter without side–effects on its performance;
- Bending die: it is responsible for the bending of some unnecessary parts that the Cutting die is not able to eliminate from the sheet metal.

These dies are basically made of pig iron melts on which other elements can be added according to the function (e.g. blades in *Cutting dies*). Moreover, holes are generally made on the melt in order to make the die lighter without hindering its functionalities.

Typically, the die designer is supported in his/her activity by CAD systems. The reference company (as well as many other manufacturing companies), adopts CATIA[2] V5.

Within CATIA, a die is considered as a "collection of parts" named features characterized by specific parameters (length, thickness, weight, and so on). CATIA supports users in the design of every part of the pig iron melt, allowing them to add semi-manufactured parts and checking for possible geometrical errors (e.g. the screw bolt is too small for allowing the screw to enter it). On the other hand, no suggestion on how a specific part of the die should be designed is supplied and this has relevant implications on the duration of a new design project. One of the first requirements derived from this lack was to exploit experiential and shared designers' knowledge for supporting this crucial decision making activity.

Starting from the end of the 1980's several tools have been developed as Knowledge-Based Engineering (KBE) tools, for example the MOKA project (Oldham et al. 1998), providing a software environment for automating repetitive engineering tasks (Sriram et al. 1989). Knowledge Based System applications have big potentials to reduce costs related to repetitive engineering tasks, but require big efforts to collect, represent and formalize the necessary knowledge. In this field one of the best known and most successful examples of application to the industrial planning of complex objects in 3D environment has been proposed by Gero and Maher (1997). They defined a conceptual and computational approach starting from the definition of design as a "*goal-oriented, constrained, decision-making, exploration and learning activity which operates within a context which depends on the designer's perception of the context"* (Gero 1990). This approach defines some knowledge representation schemes (i.e. *prototypes*) for the definition of the conceptualization and ideation process, and proposes the Case–based design paradigm to reuse previous design solutions to solve similar design problems (Maher and Zhang 1995).

The knowledge of this meaningful literature addressed the choices of IDS project on the activity of domain knowledge modeling and representation, taking into account its nature, its use and the sharing of conceptual categories, instead to force into a predefined theoretical model the knowledge coming from the field.

---

[2] http://www.3ds.com/products-solutions/plm-solutions/catia/overview/

The applied methodologically-driven (Guida and Tasso 1994), planned and intensive knowledge acquisition campaign allowed the identification of three main distinct kinds of knowledge:

- *Functional knowledge* (Scrivener et al. 2002), related to the representation of what kind of function is performed by each part of the die (e.g. the screw allows to fix the die to the press);
- *Procedural knowledge* (Friedland 1981), related to the representation of the order of steps in the design of a die (e.g. the part B should be necessarily designed after the part A);
- *Experiential knowledge*, related to heuristics coming from the stratified knowledge of the company on the domain, and increased through the experience of the professionals belonging to the Unit (but never formalized).

Functional knowledge has been captured in the IDS system by adopting an ontological model of the die (Colombo et al 2005). The die is characterized as a collection of functional systems, i.e. sets of elementary components like screws, screw bolts, holes in the melt defined on the basis of the specific function provided by the set.

Procedural knowledge has been represented by the introduction of the *SA*-Nets* specialization of Superposed Automata Networks (SA-Nets) (De Cindio et al. 1981), derived from Petri Nets Theory. Some constraints of the previous formal model have been relaxed in order to fit the features of the design domain. The main role of this procedural knowledge representation is to guide the user during the design activity, pointing out possible problems due to the violation of some precedence constraint.

Experiential knowledge has been represented as a collection of production rules modeling the heuristics followed by an expert during the design of the die, and fully integrated with the other kind of knowledge.

2.2. DIES FOR THE PRODUCTION OF CAR BODIES

In order to illustrate the main features of IDS, the *Forming die*, Figure 1, in its *simple effect* variant will be introduced.

The sheet metal is represented by the dark line in the figure. The *Forming die* is composed of a superior component and an inferior component (*upper shoe* and *lower shoe* respectively) that are moved by the press in order to obtain the final morphology for the sheet metal.

The main components responsible for the forming operation are the *punch*, the *binder* and the *die seat*, which are placed in the lower shoe

The *punch* is the die component responsible for providing the sheet metal with the required morphology in order to be used in assembling of the car body. Its geometry depends on what kind of the car body part (e.g. door, trunk, and so on) will be produced with it. The *binder* is the component of

the die that allows the sheet metal to be perfectly in contact with the punch, by blocking the sheet against the upper shoe before the punch is pushed on it. Finally, the *die seat* contains both the punch and the binder and allows the die to be attached to the press. The upper shoe of the die contains only a negative copy of the punch, usually called *matrix*.

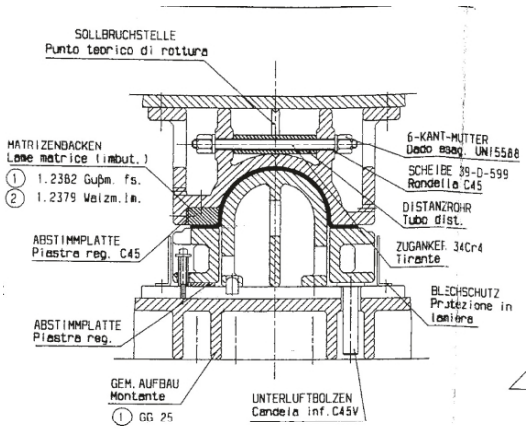


Figure 1. Sketch of the simple effect Forming die.

In case of the *Simple Effect Forming die*, the sheet metal is lifted up by the binder when the die is open. The press movement pushes the matrix down to the sheet metal and the binder. When the matrix reaches the binder, due to its opposition to the press movement, the punch is forced to move up towards the matrix, till the die is closed. When this happens, the sheet metal is pressed between the punch and the matrix, and in this way the forming operation is completed.

The design of the functional elements of the *Forming die* (punch, binder, matrix, and die seat) is the result of the complex negotiation process occurring in the group of professionals belonging to the local community of professional designers in order to fit the requirements. Next subsection will briefly describe how this community is composed and operates.

2.2. THE COMMUNITY OF PROFESSIONAL DESIGNERS

In order to obtain a die that can actually give the sheet metal the desired final shape, it is necessary to define the right sequence of operations that will be accomplished on it, before effectively manufacturing the die. The decision making process about the plan modeling involves three main kinds of actors: the *customer*, (the automotive industry requiring the final die), the *analysts* and the *designers*.

The *customer* supplies in formal way all the requirements specifying the features of the final product. In particular, the customer provides all the

useful data and information about the presses the die will be mounted on, how the sheet metal will be moved from a die to the next one and possibly some technical suggestions about how to design the different parts of the die. Moreover, the customer provides a collection of norms and constraints that should be respected during the design of the product.

Such information is elaborated by a small group of analysts, which produces a mathematical description (model) of the geometrical properties of the different parts of the die, named in the jargon simply "the mathematics". For example, the analysts define the profile of the Forming Die punch and its skin, which is a 3D elaboration of the punch profile, the dimensions and shape of the sheet metal in input to the manufacturing process and the layout of the final car body part at the end of the production process. Moreover, the analysts produce the 1/1 scale final product in the form of polystyrene model of the die shape to be studied by the designers.

Finally, Designers are responsible for the design of the die, satisfying all the customer requirements. To do this, they exploit all the information in input to the design process: in particular, the constraints of the car manufacturer, the mathematics, the layout of the involved car body parts, and the polystyrene model of the die. Then, they transform these inputs into a set of rules to be satisfied during the die design: such rules will be considered in the design of fundamental parts of the mould (i.e. the punch, the binder, the matrix and the die seat) as well as other types of principles and competences (e.g. define the punch before the binder). Designers are generally able to succeed in their tasks. However, it is possible that they have to violate some constraints according to their experience and knowledge, producing side effects on the final die shape, which will thus slightly differ from the polystyrene model made by analysts, but without hindering die performance. Otherwise, designers could ask the analysts to modify the polystyrene model, or even ask the customer to relax some constraints that they were unable to satisfy. This is a very simple example illustrating the possible contracting activity occurring in the community.
Figure 2 summarizes the actors of the professionals community involved in the design of the die for and the related interaction flow.

2.3. DIE DESIGN: PROBLEMS TO BE SOLVED

Designers' task is conceptually simple: they must translate the data represented by the polystyrene model into a CAD project to obtain a "virtual" prototype of the die. Anyway, this activity is operatively complex and time consuming, due to some main reasons.

The first reason is the need to satisfy customer's guidelines. Every customer delivers a precise operative manual containing the requirements to be taken in consideration during the project. First of all, the guidelines

concern the description of the production plant where the die will be installed: press dimensions and type (in particular, the distance between the upper and the lower parts when the press is closed); protocols for moving the sheet metal from a die to the next on; protocols for moving the die on the production line and so on. Moreover, preferences about the adoption of semi-manufactured or elementary parts are pointed out.

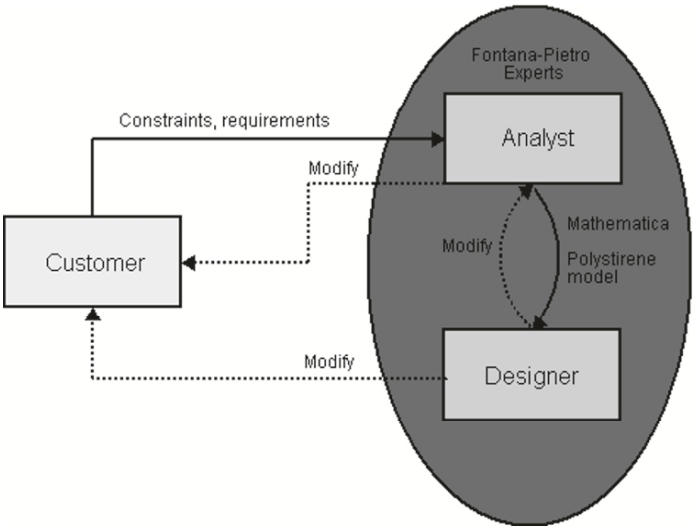


Figure 2. Schema of the involved actors and the related interaction flow.

Customer's guidelines make the design activity very long to be accomplished. Typically, the die design could require up to fifteen professional designer's working days. This production time can be significantly augmented in case of design errors.

Another important limitation to the die design comes from the fact that some standard methods to tackle the design of dies is not shared: every designer has his/her own style, and in general it is unlikely that a designer continues a project started by a colleague (and this is considered an extremely difficult task). In case of absence of a designer, the project followed by him/her is suspended. Another consequence is that designer's shared experience and knowledge are often tacit, because of the nature of professionals composing the community of professionals.

The difficulties in building a common and shared know-how about the die design activity has also a relevant impact on the learning process by newcomers. Young designers observe more expert colleagues in their daily activities for a given time period, whose duration varies according to the

capabilities of the expert. Doing so, newcomers are motivated to learn the basic aspects rather than the deep know-how of the die design activity.

In order to solve these problems and build an effective tool for supporting die designers in their decision making process, the development of a knowledge-based system taking into account designers competences can be the right choice. To do this, the company management allowed interacting with some of the most expert designers in the community, in order to capture the knowledge involved in the decisional process and build a complete, shared and unique knowledge model.

The knowledge engineering activity led the choice of focusing on the peculiarities of the different decisional and interaction roles played in problem solving. Different kinds of knowledge structures and inference methods were found, in accordance to different goals to be reached during the design process. The embodiment of methods coming from economical organizational and social sciences taking into account the role of the interactions and the value of shared knowledge into the Knowledge Engineering methodological framework (Bandini and Sartori 2005) enhances the discovery of knowledge structure for problem solving, and shows new perspectives both in knowledge engineering and knowledge representation.

This means that the use of off the shelf and uniform implementation solutions is not the best choice. It is the case of rule-based systems. Forcing in a unique representation formalism the heterogeneity of knowledge structures in this case was simply wrong and not "natural".

Although it is possible to directly implement a Rule-based system by the exploitation of a specific module of CATIA (in the *Knowledgeware* release), the above methodological reasons and the general lack of available documentation on its internal mechanisms led to the development of an original solution, designing and implementing the knowledge-based module (IDS) from scratch. This module directly interacts with CATIA, exploiting the complex knowledge structures captured during the knowledge engineering and knowledge representation activities.

3. Supporting the Design and Manufacturing of Car Body Dies

The intensive (4 months) knowledge acquisition campaign revealed three different kinds of knowledge: functional, procedural and experiential. In the following, how each of them has been conceptualized and represented will be given.

3.1. FUNCTIONAL KNOWLEDGE: TOWARDS THE "DIE ONTOLOGY "

The first shared knowledge structure allowing the contracting activity in common problem solving can be represented using the Ontology approach

(Guarino 1995). At a first glance, the hierarchical structural decomposition of the die (*is-a*, *part-of* relations) could represent the right structure of this kind of knowledge, because of the classificatory capabilities of the senior design professionals. The mere joining of this kind of ontological set-up with knowledge involving the functionalities (not captured by *is-a*, *part-of* relations) of the involved mechanical parts can be conceptually complicated and sometimes not feasible. A different and more suitable conceptualization has been adopted as shown in Figure 3.

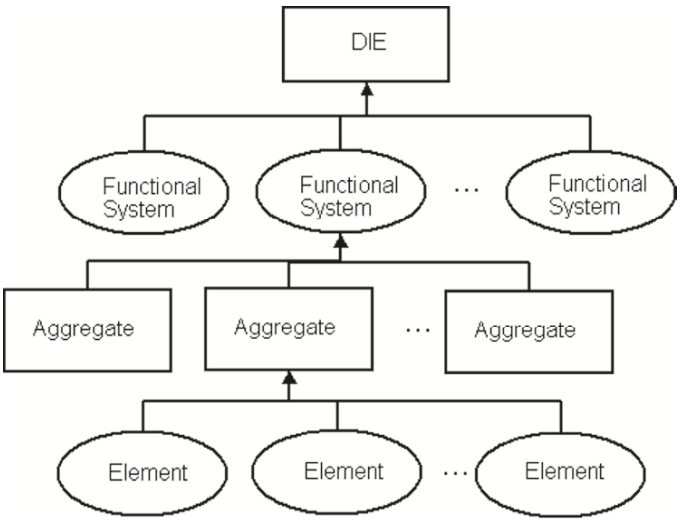


Figure 3. A die can be decomposed according to three different levels of complexity, into several functional systems, aggregates and elements.

The die is requested to perform different functions. For example, forming die must provide the sheet metal with the desired initial morphology (changeable during some next step), and this function will be guaranteed by a specific group of die elements. But the forming die must also be moved from a press to another one, and the *movement-ability function* will be accomplished by another group of parts. Each conceptual part of the die that performs a specific function is called functional system. A die can then be considered as a collection of one or more functional systems.

Functional systems, however, can be fairly complex. Sometimes, designers conceive them as a composition of lower level Aggregates, which are semi-manufactured components that can be grouped together in order to make simpler and faster the design of a Functional Systems.

Finally, Elements are instead (atomic) elementary parts (screws, for instance). Their role can be different according to the aggregate (and thus functional system) they belong to.

As an example of this categorization of the functional system, consider the fixing function. The die must be mounted on a press: for this reason a specific functional system, named fixing system, schematized in Figure 4 part A, should be provided.

This system can be designed in two different ways, both accordingly to customer requirements, based on screws, Part B in Figure 4, or dowels, Part C in Figure 4.

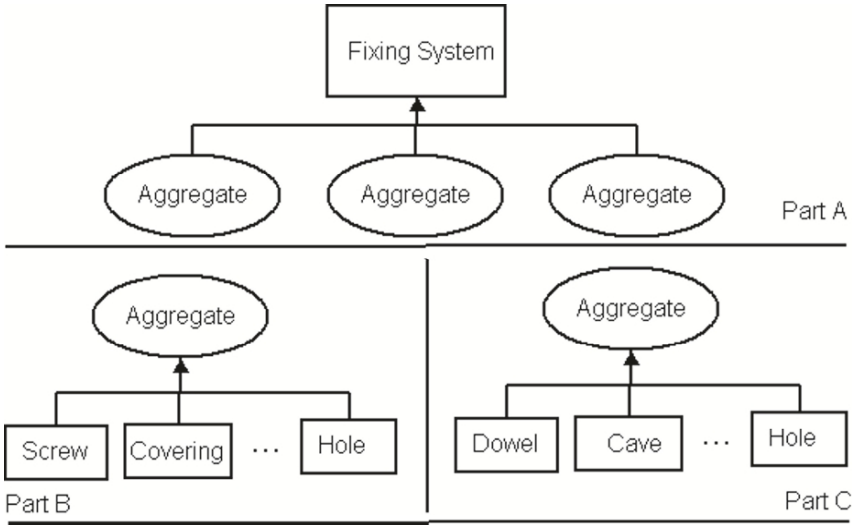


Figure 4. The categorization of a die fixing system as set of aggregates (Part A) that can be based on screws (Part B) or dowels (Part C). Functional systems and aggregates are bounded by part-of relations, as well as aggregates and elements.

This categorization is fundamental for all the knowledge representation choices, being the main ontological conceptualization of the domain, permitting IDS to represent a die on the basis of the function it will perform (according to the designer way of thinking) rather than of the elementary parts it is made up of (the vision of a CAD system like CATIA).

3.2. PROCEDURAL KNOWLEDGE: SA*-NET

The introduced ontological model is not sufficient to give an exhaustive representation of the knowledge involved in die design activity. Each functional unit described in the ontology should be designed according to a specific set of procedural constraints to be represented in the proper knowledge representation schema.

The representation of procedural knowledge is a classical issue in Artificial Intelligence (Winograd 1975), and many studies on this topic have

been developed (Georgeff and Bonollo 1983). With reference to the literature concerning this topic, in the IDS project our focus can be set in Petri-nets based formalisms to represent procedural knowledge. In particular we considered Event-Graphs (Gallanti et al. 1985) as a starting point. However, due to the nature of the domain, the process centered characteristics of Event-Graphs didn't allow to represent the complexity of the adopted ontological structure. Then, we considered Superposed Automata Networks (SA-Nets) formalism (De Cindio et al. 1981) a sub-class of Petri Nets previously defined in the area of languages for the analysis and design of organizational systems, and the study of non-sequential processes.

This solution, coupled with ad hoc modifications brought us to the definition of a new SA-Nets class, namely SA*-Nets (Colombo 2005). The most relevant feature of SA*-Nets to the IDS context is the possibility of defining, for each ontological element of the die, the design activity steps it is involved on.

A SA*-Net is a graph made of set of nodes and labeled transitions. Nodes permit to trace the current state of the project, while transitions identify the different design steps. Two different classes of transitions have been considered in the design of SA*-Nets:

- Descriptive transitions, that are labeled with the name of a functional system considered in the die ontology, permit to pass from the description of a part to the related design process;
- Design transitions specify all the design steps necessary to complete the definition of the corresponding descriptive transition.

Figure 5 shows a sample SA*-Net, where it is represented a sketch of a die part (i.e. die seat, punch, binder or matrix) as a set of descriptive transitions (boxes with round corners in the figure) having the name of a functional system defined by the die ontology. Each descriptive transition is linked to one or more design transitions (boxes in the figure) allowing to define how that functional system is configured. Design transitions permit to describe aggregates and elementary parts of the die ontology. For example, in the case of a descriptive transition Fixing System we would have a sequence of design transitions for choosing the type of screws to be used, the dimension of the corresponding holes, and so on.

Unlike traditional SA-Nets, SA*-Nets are characterized by a semantic completely defined by its transitions; in fact, while in the SA-Nets nodes act as *tokens*, with the consequence that a transition can be activated if and only if all its entering nodes are marked, in the SA*-Net the only function of a node is to trace the next part of the die to be designed. In this way, a designer can decide which part to define, being sure that the system will be

able to support him/her in the execution of all the steps necessary to complete the chosen part of the die.

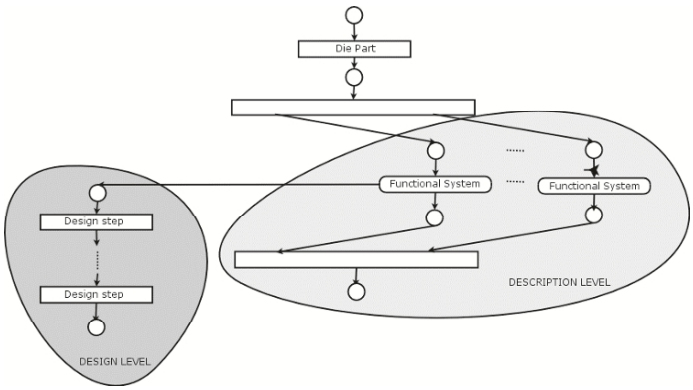


Figure 5. A SA*-Net has two classes of transitions, descriptive transitions and design transitions.

Since the set of design activities is composed of a number of steps that are not necessarily sequentially ordered, the SA*-Nets are provided with syntactic elements to manage sequential, concurrent and binding processes, as shown in Figure 6:

- A *sequential process* is a collection of design steps that must be necessarily accomplished according to a sequential order;
- A *concurrent process* is a collection of design steps that can be executed at the same time;
- A *binding process* is a collection of design steps belonging to *different* descriptive transitions where the execution of the transitions must preserve specific precedence constraints. While the first two compositions are the basic tools to build single part design processes, the latter allows the specification of relations among design processes of different parts.

While SA*-Net syntax inherits from SA-Nets syntactic elements to deal with sequential and concurrent processes, the *fork/join* transitions and *or/and* nodes, Figure 6(a) and 6(b), the management of binding processes has requested to enrich the SA*-Nets by including the possibility of representing *constraints* between two subnets, Figure 6(c). Constraints can link design transitions of different descriptive transitions, helping the designer in preventing possible negative side-effects in the design of die parts.

In Figure 6(c), the two sequences of processes can be executed at the same time, but the dotted arc linking step 1 of the second sequence and step 2 of the first one means that undesirable side effects could emerge in case of accomplishing step 2 before step 1.

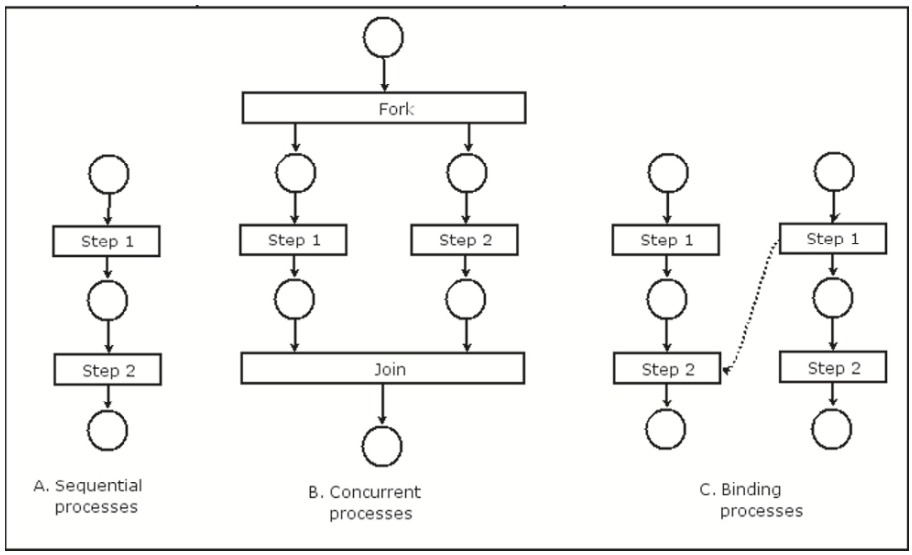


Figure 6. (a) Sequential processes in the IDS SA*-Net; (b) concurrent processes in the IDS SA*-Net. The synchronization is given by the fork and join transitions; (c) Binding processes in the IDS SA*-Net.

A concrete example of these possible side effects is shown in Figure 7, since the binder profile is adjacent to the punch one, the binder should be generally designed after the punch, as in Figure 7(a) of the picture. However, a designer could decide to describe the binder first. In this case, possible side-effects like the one drawn in Part (b) of the picture could happen, where the punch dimensions exceed those of the binder. IDS can then warn the user.

Four SA*-Nets have been designed and implemented in IDS, one for each part of the die: Die Seat net, Matrix net, Binder net and Punch net.

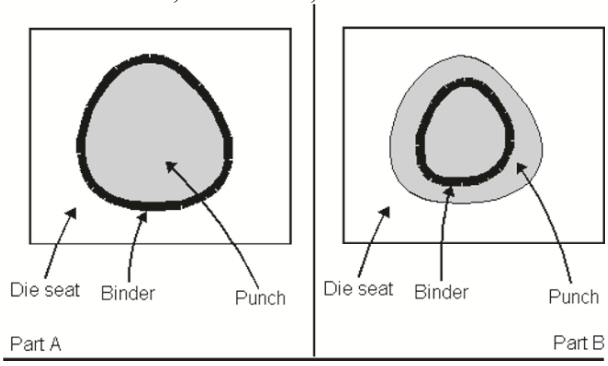


Figure 7. (a) the binder has been correctly designed after the punch, since the punch must slide inside it; (b) the binder has been defined before the punch, with a violation of geometrical constraints.

3.3. EXPERIENTIAL KNOWLEDGE: THE DESIGN RULES ADOPTED BY EXPERTS

Finally, a very important aspect of the die design decision making process captured by the IDS system was the possibility for an expert designer to exploit his/her own experience to execute a specific task.

SA*-Nets are able to capture procedural aspects in the design of a functional system, but they cannot be used to evaluate its parameters and geometric features. The configuration of a functional system in terms of height, width, weight and so on is made directly by a set of rules representing related designer's competence. In other words, a set of rule represents a way of navigating the SA*-Nets according to the competence of an expert: when a designer asks IDS to be supported in the design of a specific descriptive transition (i.e. a functional system), the SA*-Net specifies all the design transitions to be accomplished. Moreover, a corresponding set of rules for each of them is activated to evaluate all the functional system attributes or to give suggestions about the right position for a given part in the current state of the design, Figure 8.

A rule can be activated if all the preconditions (i.e. the left hand side) are true. A precondition in the IDS project is generally a test of the presence of a given initialization value (e.g. the type of customer's press) for a part or the satisfaction of a constraint (e.g. the dimensions are smaller than a threshold).

Initialization values are startup information about the project (e.g. who is the customer, what kinds of presses the die will be mounted on and their dimensions, and so on). Such information can induce modifications in the specification of the design transition: for example, a customer could impose to designers the use of dowels instead of screws in the definition of Fixing System. This fact would be represented in the IDS system by the definition of two distinct rules, as shown in Figure 9.

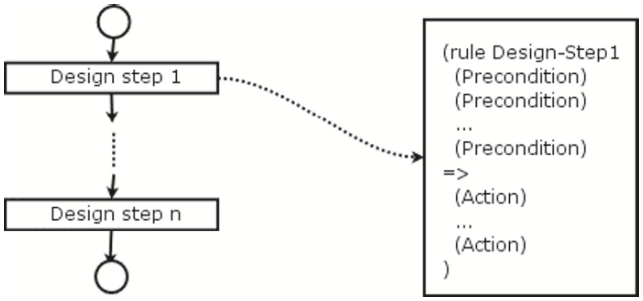


Figure 8. When a specific design step of the SA*-Net is reached, one or more rules are activated to execute all the actions necessary to complete it.

Initialization values are startup information about the project (e.g. who is the customer, what kinds of presses the die will be mounted on and their

dimensions, and so on). Such information can induce modifications in the specification of the design transition: for example, a customer could impose to designers the use of dowels instead of screws in the definition of Fixing System. This fact would be represented in the IDS system by the definition of two distinct rules, as shown in Figure 9.

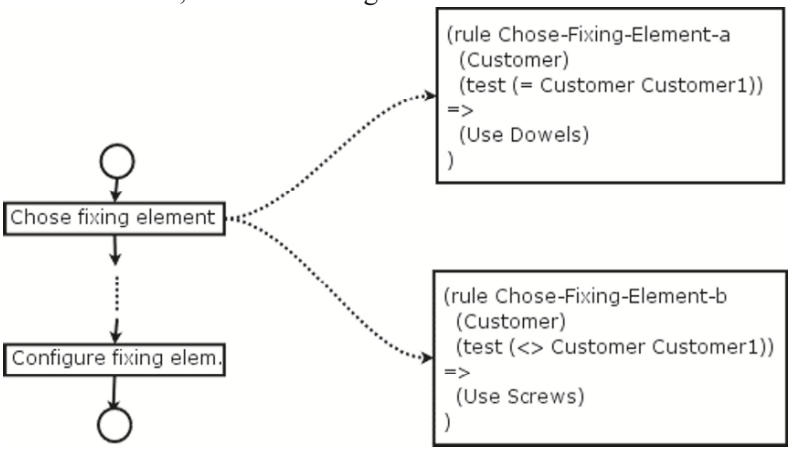


Figure 9. The same design step could be specified by different group of rules according to different preconditions. Here, the choice about the use of dowels or screws in building the Fixing System depends on the name of the customer.

Preconditions in the left hand side of a rule can also be the specification of a constraint between a binding process and another design step. In this case, the binding process should have been executed before the other in order to generate useful information for the second one (e.g. the dimension of a hole is useful for choosing the right screw). An example of this kind of constraints is the creation of an object of the ontology: the binder is typically designed after the punch because its width and length are equal to the ones of the punch. Thus, there is a constraint between the punch and binder SA*-Nets such as the one shown in the first part of Figure 10.

When the designer is going to define the binder width and length, the existence of a constraint involving the corresponding design transitions coming from the punch SA*-Net is detected. This constraint is specified by a rule through the specification of a test about the precedent evaluation of punch width and length. If the test is satisfied (i.e. there exists an ontology element named punch that has been created as a consequence of the punch descriptive transition in the punch SA*-Net whose width value is different from null) the binder width can be evaluated by IDS. Otherwise, the user will be notified about the need for executing the define width design transition in the punch SA*-Net before proceeding with the binder.

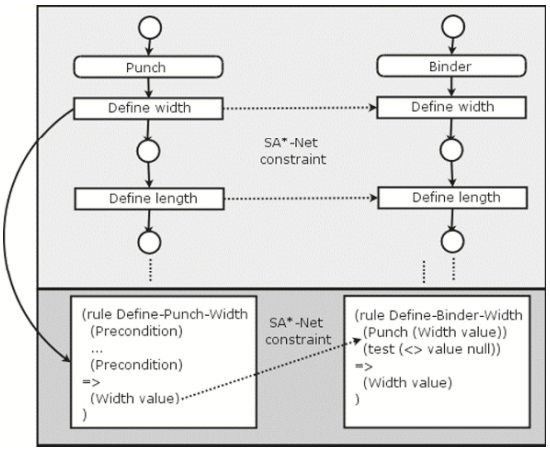


Figure 10. How to represent constraints between design transitions in the corresponding rules.

4. IDS Implementation

Figure 10 shows a sketch of the high level system architecture. The IDS system is a collection of knowledge–based and communication modules that interacts with CATIA 5.0, the CAD tool used by expert designers of Fontana–Pietro in their daily activities. The system has been implemented exploiting the client-server architecture, where CATIA acts as the client and IDS as the server. The system is made up of three logical components: the knowledge-based module, the CATIA-IDS connector and the knowledge repositories.

There are three knowledge repositories, one for each type of knowledge identified: a collection of Java objects, a collection of XML files and a collection of production rules. Java objects implement the IDS ontology: every part of the die has been represented, starting from the functional systems up to elementary components. XML files have been adopted for the implementation of the SA-Net to describe procedural knowledge as well as the SA- Net Manager, a software module that allows browsing the SA-Net and managing it by adding new states, transitions, constraints and so on. Finally, a collection of files containing rules for implementing experiential knowledge has been integrated in the knowledge base.

The knowledge based modules communicate with CATIA through the ad-hoc developed software module called *Catia-IDS connector.* Although CATIA promises an easy interconnection by standard mechanisms like CORBA, we have verified that it is not simple to use these functionalities, due to the difficulties in obtaining useful documentation. Thus, we have

decided to make CATIA and IDS communicate through a TCP socket connection that is managed by CATIA. An opportune syntax has been thought for building messages between CATIA and IDS (a message contains at least the name of the required service, a list of parameters to be valued) and vice versa. To allow the communication between CATIA and IDS, an extension of CATIA has been made by *Fontana Pietro* R&D department, with the creation of a suitable GUI.

A die is considered by CATIA as a collection of components, named *features*. Features are geometrical templates where the shape is characterized but no numerical values to dimensional parameters are defined. Thus these parameters must be evaluated by the designer in order to let the feature to become effective parts of the die. While CATIA doesn't support the designer in valuing the different parameters of a feature to be inserted in the die project, IDS has been designed and implemented to help the designer in valuing feature attributes.

For example, an IDS session starts when the designer asks CATIA for a new feature insertion, i.e. a new operation that modifies the status of the project. The communication between IDS and CATIA is guaranteed by a specific communication protocol exploiting TCP socket connections. IDS receives from CATIA all the information about the current state of the die project (e.g. what kinds of parts and products have been already designed, which are the values of their attributes and so on), as well as the general information about the project (e.g. the customer and its presses, the mathematics coming from analysts, and so on). Then, IDS instantiates all the

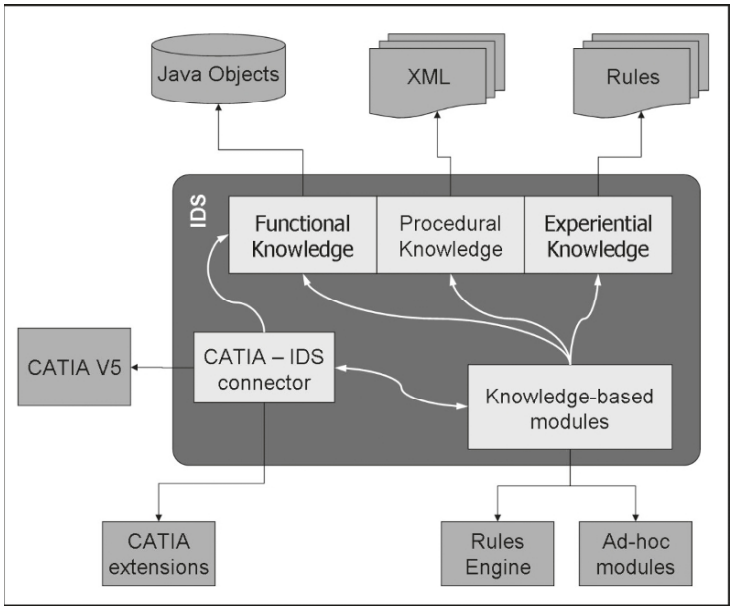


Figure 10. The high level architecture of the IDS system.

objects necessary to its functioning: each object is a Java implementation of an ontological part of the die that has to be evaluated. Afterwards IDS starts the elaboration by activating the related SA*-Net and corresponding rules.

5. Conclusions and Future Works

The IDS system has been designed and implemented to support experts of *Fontana Pietro* in the design of dies for the production of car-bodies.

The IDS system required about twelve months to be developed. It was part of a research project funded by Italian Minister for Research and Development where *Fontana Pietro* was project leader. The IDS system was delivered last June and it successfully passed the examination by the Italian Minister for Research and Development.

Currently, only the knowledge base related to Forming Die has been designed and implemented. A prototype of SA*-Net and rules for the Cutting Die has been developed too, but it is not working due to the absence of necessary CATIA extensions.

Two main results have been achieved:

- A complete model (Bandini et al. 2005) of the knowledge involved has been created, that allows to extend the functionalities of CAD tools, with benefits for designers from the project timing point of view;
- Knowledge sharing within the expert community of *Fontana Pietro* is facilitated (Colombo et al. 2005), since the IDS project has allowed to unify into a unique knowledge model, understandable by all *Fontana Pietro* designers.

Possible developments of the IDS system concern its linking to a FEM module, for the quantitative analysis of the designed die. The main objective of such extension should be the evaluation of structural properties of a die designed by CATIA exploiting the services of IDS in order to understand if it will work correctly or not. In this way, the FEM would receive a 3D model of the die from CATIA, then it will analyse the die exploiting mathematical models and return positive or negative results of the evaluation of initial requirements satisfaction to IDS.

Then, IDS could start a new elaboration taking care of FEM report; obviously, an extension of the currently implemented knowledge base should be made that takes in consideration how to adapt a project to solve errors during the first design phase. In this way, another important and expensive phase of the die cycle of life could be partially automated: the die testing. Currently, the test of a designed die is made through a prototype of the final product. All the functionalities of the die are tested: in particular, the result of a forming operation on a sheet metal is evaluated to verify if it is correct or not. In the second case, it is possible to eliminate manually small imperfections, but in case of bigger problems the project has to be

restarted, with significant time and money losses (the pig-iron must be discarded).

The interaction between IDS system and a FEM module could prevent big problems by simulating the behavior of the die avoiding the physical production of a bad prototype. At the moment, we are verifying with the Management of *Fontana Pietro* the possibility to extend the IDS project to take care of this new functionality.

Acknowledgements

Authors acknowledge Fontana Pietro people for their support in the Project. Special thanks to Fabrizio Tagliabue, Salvatore Monreale and Marco Bertoldini.

References

Bandini, S, Colombo, G, and Sartori, F: 2005, Towards the integration of ontologies and SA–nets to manage design and engineering core knowledge, *in* MA Sicilia (eds), *Electronic Proceedings of ONTOSE 2005*, Alcalá de Henares.

Bandini, S and Sartori, F: 2005, CKS–Net, a conceptual and computational framework for the management of complex knowledge structures. *The 2nd Indian Conference on Artificial Intelligence*, Pune, India.

Colombo, E, Colombo, G and Sartori, F: 2005, Managing functional and ontological knowledge in the design of complex mechanical objects, *The 9th International Congress of Italian Association for Artificial Intelligence*, LNAI 3673, Springer-Verlag, Berlin-Heidelberg, pp 608-611.

Colombo, G: 2005, *Representing and Managing Designer and Engineering Core Knowledge: Ontologies and Engineering Core Knowledge*, PhD Dissertation, University of Milan-Bicocca, Milan, Italy.

De Cindio, F, De Michelis, G, Pomello, L and Simone, C: 1981, Superposed automata nets, *in* C Girault, W Reisig, (eds), *Application and Theory of Petri Nets, Selected Papers from the First and the Second European Workshop on Application and Theory of Petri Nets*, Stasbourg, pp. 23-26.

Friedland, P: 1981, Acquisition of procedural knowledge from domain experts. *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 856-861.

Gallanti, M, Guida, G, Spampinato, L, Stefanini, A: 1985, Representing procedural knowledge in expert systems: An application to process control, *Proceedings of Ninth International Joint Conference on Artificial Intelligence - IJCAI 1985*, Los Angeles, California, pp. 345-352.

Georgeff MP and Bonollo, U: 1983, Procedural expert systems, *Proceedings of IJCAI 1983*, Karlsruhe, Gernmany, pp. 151-157.

Gero, JS and Maher, ML: 1997, A framework for research in design computing, *in* B Martens, H Linzer and A Voigt (eds), *ECAADE'97, Osterreichischer Kunst und Kulturverlag, Vienna (CD-ROM)*, Topic 1, paper 8.

Gero, JS: 1990, Design prototypes: A knowledge representation schema for design, *AI Magazine* **11**(4): 26–36.

Guarino N: 1995, Formal ontology, conceptual analysis and knowledge representation. *Int. J. Hum.-Comput. Stud.* **43**(5-6): 625-640.

Guida G and Tasso C: 1994, *Design and Development of Knowledge-Based Systems: from Life Cycle to Development Methodology*, John Wiley, Chichester, UK.

Nishimoto K, Riva J, Vega Perez RM, Vassallo M and Camilli A: 1991, CATIA and CAEDS applications for ship and ocean system design, *Computer Applications in the Automation of Shipyard Operation and Ship Design*, pp. 61-74.

Oldham, K, Kneebone, S, Callot, M, Murton, A and Brimble, R: 1998, MOKA - A methodology and tools oriented to knowledge-based engineering applications, *in* N Mårtensson, R Mackay and S Björgvinsson (eds), *Changing the Ways We Work, Advances in Design and Manufacturing, Volume 8, Proceedings of the Conference on Integration in Manufacturing*, Göteborg, Sweden, IOS Press, Amsterdam, pp. 198-207.

Scrivener, SAR, Tseng, WS-W, and Ball, LJ: 2002, The impact of functional knowledge on sketching, *in* T Hewett and T Kavanagh (eds), *Proceedings of the Fourth International Conference on Creativity and Cognition,* New York: ACM Press.

Sriram S: 1989, *Knowledge-Based System Application in Engineering Design Research at MIT*. MIT press.

Winograd, T: 1975, Frame Representations and the Declarative/Procedural Controversy. *Readings in Knowledge Representation,* Morgan Kaufman, pp. 185-210.

Zhang, DM and Maher, ML: 1995, Case-based reasoning for the structural design of buildings, *IEA/AIE 1995*, pp. 141-150.

DESIGN METHODOLOGIES

System development methodologies: A knowledge perspective
Warren Kerley and Tony Holden

Analogical matching using device-centric and environment-centric representations of function
Greg Milette and David Brown

Design operators to support organisational design
Catholijn Jonker, Alexei Sharpanskykh, Jan Treur and Pinar Yolum

Bayesian design networks
Peter Matthews

SYSTEMS DEVELOPMENT METHODOLOGIES: A KNOWLEDGE PERSPECTIVE

WARREN KERLEY AND TONY HOLDEN
Cambridge University, UK

Abstract. Structured methodologies have for some time been the dominant guiding means by which information systems have been designed and implemented. This paper argues that the process-based view that has underlied information system design (ISD) for so many years could usefully give way to a knowledge-based perspective since ISD is fundamentally a knowledge structuring activity. This paper includes a review of the evolution of systems development methodologies since the late 1960s and contrasts the intended benefits and problems in practice of using structured, process-oriented methodologies such as SSADM, with those of the agile methodologies, such as XP, which have emerged in the last few years. A framework is presented of how ideas from the fields of knowledge management and organizational learning can be applied to the analysis of ISD methodologies. We consider some higher-level considerations of possible future trends in ISD and our analysis suggests that the present preponderance of structured methodologies will give way to greater use of agile and open source approaches. We finish with a description of the work we have done to apply these ideas within current practice and make suggestions for further work in this area.

1. Introduction

Developers of computer-based information systems (IS) are expected to deliver working systems that meet their customers' requirements and are constructed to acceptable levels of quality, maintainability, dependability, efficiency and usability. They are also expected to do this in a way that is cost effective, timely, manages the levels of risk to which the development project is exposed and is flexible enough to incorporate changes throughout the project lifecycle.

In practice this is difficult. It is an established part of ISD wisdom that many systems are delivered late, over budget and behind schedule. Others are cancelled without ever being completed. Even if a system is delivered within time and budget constraints, it may still be perceived as a failure if

J.S. Gero (ed.), Design Computing and Cognition '06, 163–181.
© 2006 *Springer. Printed in the Netherlands.*

does not meet the expectations of key stakeholders. Moreover, the productivity of software developers has failed to keep pace with the spectacular fall in cost and increase in performance of computer hardware.

2. Methodologies

Researchers and practitioners have not been lax in meeting this challenge. Methodologies have proliferated over the last 30 years in order to organize the activities involved in information systems development. In the 1960s and 70s information systems were developed largely on an ad hoc basis, without any formal methodological support. Driven by the need to manage increasingly large and complex projects, efforts were made to formalize matters and, in the late 1960s, the systems development lifecycle (SDLC) view of software project development, commonly known as the waterfall model, emerged. Many methodologies follow the SDLC approach today due to the support of national governments and organizations such as the US Project Management Institute. Nevertheless by the late 70s and early 80s new approaches based on prototyping and evolutionary development had appeared in response to practitioner experience that showed SDLC methodologies could take too long and be too rigid under some circumstances.

Other work by academics in the early 80's aimed to address the social and sense-making issues involved in ISD and resulted in, for example, ETHICS and the Soft Systems Methodology (SSM). Subsequent work combined the best elements of the existing methodologies, such as merging of prototyping and evolutionary development ideas with the SDLC to produce iterative and spiral methodologies.

The proliferation of methodologies reflected a number of different – and rarely stated – philosophies, assumptions or beliefs about the nature of ISD. These can nevertheless be broadly classified in a number of different ways. For example Avison and Fitzgerald (2003) suggest seven major themes or approaches – structured, data-oriented, prototyping, object-oriented, participative, strategic and systems – which are not necessarily mutually exclusive.

Although supplemented in recent years by visual or object-oriented methods for interface design and specialized systems, the majority of formalized methodologies in the 1990s were still underpinned by structured SDLC and/or prototyping process models. They are therefore still primarily process-oriented and plan-driven with their methods and process models based on concepts that originate in mainstream engineering disciplines. Projects are heavily documented and require strict adherence to the prescribed processes.

Despite all this being highly time and effort consuming, documentation does aid in realizing a degree of quality, control and, importantly for most software companies, reassurance for potential and repeat customers.

Organizations using these approaches consequently look for ways to continuously improve these processes through adherence to quality standards such as ISO certification and process improvement initiatives such as SEI's Capability Maturity Model (CMM).

From a management viewpoint, subdivision of the development process has definite advantages. It reduces the skill-levels required by developers and standardization makes these skills more interchangeable. These are seen as critically important benefits given the shortage and high turnover of software developers (Riemenschneider 2002). Standardization also facilitates project management and control, thereby reducing risk and uncertainty. The large amount of documented information within the process increases management flexibility, as personnel can be moved quickly within or between projects, and provides insurance against the loss of critical knowledge if key personnel leave. These benefits have prompted national governments to encourage the use of structured methodologies and the subsequent support of formal certification and standards such as ISO 9000, ISO 12207, ISO 15504, SSADM within the UK and the CMM by the US Department of Defense.

In practice, however, the use of formalized, process-oriented methodologies has not been without problems. In a case study of a project using SSADM, Wastell (1996) found that the methodology was followed in a blind, mechanical way. The effort spent on the aesthetics of the diagramming techniques, and in providing the detail required by the documentation standards, bogged the project down and provided too much information. The "big picture" was obscured as far as the user representatives were concerned. Process-oriented methodologies work best when the requirements of the software project are completely locked in and frozen before the design and software development commences. However, in an increasingly volatile business environment, firms are asking for lighter weight, faster and more agile software development processes that can accommodate the inevitable ongoing changes to requirements. At the same time there has been a backlash from the programming profession against the mechanistic and dehumanizing aspects of process-oriented methodologies and a desire for a return to programming as a craft rather than an industrial process. These factors have led to the development by practitioners of agile methodologies in the late 1990s, based on prototyping and rapid application development approaches (Abrahamsson et al. 2002).

eXtreme Programming (XP) has been widely acknowledged as the starting point for the various agile software development approaches and is probably the best known. Other agile methodologies include Scrum, Dynamic Systems Development Method, Crystal Methods, Feature-Driven Development and Adaptive Software Development.

No clear agreement has been achieved on how to clearly distinguish agile software development from more traditional, process-oriented approaches. A central tenet of the agile philosophy is that it is impossible to get software

right first time and it is therefore preferable to be responsive to changing customer requirements and to provide them quickly with what they want. Common, although idealized, features of the agile approach are that software development is:

- Incremental with small functional differences and shorter times between releases.
- Cooperative with customer and developers working constantly together with close communication,
- Straightforward in that the method itself is easy to learn and to modify,
- Adaptive and so changes to requirements may easily be included,
- Well, but minimally, documented.

It is this last aspect that most characterizes agile methodologies. It could also be argued that agile methodologies are "more honest" as their approach mirrors more closely the reality of software development in practice.

Concerns have been raised about the use of agile methodologies (Abrahamsson 2002; Boehm and Turner 2003). They do not provide the familiar management control mechanisms and high quality assurance inherent in process-oriented methodologies, and are therefore considered risky. There are also serious doubts about how scalable agile methodologies are to larger projects. Nevertheless experience has shown that both process-oriented and agile methodologies have a role in contemporary software development.

That said how should a project manager choose which methodologies or methods to adopt? No single methodology can work for all types of project. Guidance is needed for practitioners about the methodologies or specific methods that are applicable in a particular circumstance and how to select the best one. The early view that there might be a single best method for all IS development has given way in recent years to investigation of domain-specific methods (Barry and Lang 2002). Although a single methodology may be appropriate in some circumstances, no methodology covers all aspects of systems development and each methodology tends to be stronger on some aspects than others. Many problem situations will require developers to use methods taken from different methodologies. However it requires considerable skill on the part of developers and managers to be able to pick and choose between methods and apply them effectively to the task in hand.

Besides the problem of selecting from the vast numbers available, methods from different methodologies may be incompatible with each other because they are based on different philosophical assumptions or emphasize different modeling stances and therefore representations of the system being considered. For example, Stephens and Rosenberg (2003) describe in detail the dangers of using some, but not all, of the twelve practices of eXtreme Programming.One solution is a methodological framework such as Mulitview that provides a coherent method to choose appropriate methods,

tool and techniques contingent on the problem, the methodology and the information systems development team itself (Avison and Wood-Harper 1991). More recently Boehm and Turner (2003) have outlined a contingency method to identify the parts of a project that are amenable to agile methodologies and those suitable for a process-oriented approach.

3. Use and Benefits of Methodologies

After all the theorists and academics have had their say, are methodologies actually used in practice? Many companies do not use commercially available methodologies at all or extensively modify them to better fit their own particular organizational needs. The scale of this phenomenon means that in-house methods predominate over the formalized methods prescribed in the literature (Barry and Lang 2002). Ironically, little is known about the nature of "homegrown methodologies" or how they are developed.

Where formal methodologies are used, they are rarely followed closely. For example, the pressure of deadlines often leads to practices being modified or abandoned for the sake of expediency (Wastell 1996; Curtis et al. 1988). Estimates vary but between 50 and 75 per cent of US organizations would be classified at CMM Level-1. Namely, immature software organizations in which development is inconsistent and methodologies are not used (Riemenschneider 2002). In one UK survey 60 percent of respondents did not use a development methodology and only 14 per cent claimed to use a formalized commercial methodology, Table 1. "The predominant reason for non-use cited by respondents was that currently available methodologies did not suit the profile of the development prevailing in the organizations studied." (Fitzgerald 2000).

TABLE 1. Methodology Usage.

	%
Organizations not using any methodology	60
Organizations using a formalized commercial methodology	14
Organizations using internal methodology based on a commercial one	12
Organizations using internal methodology not based on a commercial one	14

Secondly, do methodologies work? Adopting a new methodology is a costly and radical step. It involves significant organizational changes; substantial investments in technology, training and staff time; and the need to overcome resistance to change by developers and other stakeholders (Abrahamsson 2002; Riemenschneider 2002; Wastell 1996). Barry and Lang (2002) found that information systems developers appear to be reluctant to abandon older techniques, even when their usefulness may be questionable, and are slow to adopt the new techniques. In this context it seems unlikely

that IS departments will want to invest in multiple methodologies unless they are likely to be effective.

Unfortunately most work to date has been focused on developing new methodologies rather than evaluating their efficacy in practice. Where empirical research has been done the results can be equivocal or contradict other studies. Glass (1999) reviewed the research on a number of new technologies – including structured methodologies – that were expected to bring significant improvements in software development productivity. Fourth generation languages (4GLs) and object oriented (OO) approaches seem to have provided the biggest productivity improvements, although Glass has reservations about endorsing them.

As for structured methodologies, despite their longevity, research into their benefits was surprisingly scarce and could point to, at best, modest benefits from their use. The same lack of hard empirical evidence for their benefits applies to agile methodologies (Abrahamsson et al. 2002), although this may clearly be due, at least in part, to their relative newness. The case studies that have been published on the use of agile methodologies have often shown spectacular improvements in productivity and quality, but these results have been greeted with skepticism by other practitioners and academics who cite the absence, to date, of sufficient data (Boehm and Turner 2003). Third, and finally, are methodologies relevant for the future? Formalized, process-oriented methodologies were seen as the solution to the "software crisis" but have not delivered the benefits expected (Glass 1999). Structured and agile methodologies are based on concepts that came to prominence in the decade between 1967 and 1977 Since then the pace of business change has increased significantly, short-term needs dominate and the economic justification for formalized systems development with its long development lifecycle is dwindling (Boehm and Turner 2003). Systems development is increasingly outsourced or based on the customization of packaged software.

And it has been long recognized that methodology is less important than the skill and determination of developers. The majority of methodologies in use follow a rational, scientific paradigm where information systems development is conceptualized as an orderly process that is amenable to the same sorts of methods as mainstream engineering. In fact practice shows that systems development is anything but rational and orderly and often too little attention is paid to 'softer' social aspects and to human factors such as creativity, intuition and learning over time.

In summary, IS managers are faced with a wide choice of possible methodologies and, although contingency approaches to select methodologies have been proposed, this choice has been further complicated by the agile vs. process-oriented debate. In some respects however the academic and practitioner literature gives greater importance to the formalized, published methodologies than their use in practice warrants, as they are often either not used or are heavily customized in practice. There

are also doubts about their current efficacy and their usefulness in the future given the rapidly changing business environment.

It must be acknowledged however that the use of structure and formality over the last couple of decades has contributed in some way to the delivery of large, complex and functioning software systems. Rather than invent yet another methodology, the following sections take a different, knowledge-oriented perspective on the task of software development. From this view how can the ideas from the knowledge management and organizational learning literatures contribute to a better understanding of the practice of ISD and therefore which methods, tools and techniques should be most effective in delivering IS projects in any given organizational situation?

4. A Knowledge Perspective

Brooks (1987) identified four essential difficulties (or essences) inherent in developing software that make information systems development different from mainstream engineering disciplines:

1. *Complexity:* Software systems have a very large number of different states that increase more than linearly with an increase in system size. This complexity creates problems of communication and understanding, testing and verification, use, reuse, modification and maintenance.
2. *Conformity:* Software is generally expected to conform to the needs of the organization and not vice versa.
3. *Changeability:* Software is highly malleable and therefore successful software will be changed, either at user request or to be used elsewhere.
4. *Invisibility:* Implemented software is invisible. It is also difficult to visualize as multiple modeling techniques are required to fully represent its function and structure. There is also no representational single point of reference. There is no software equivalent of the floor plan of a building or the circuit diagrams and mechanical drawings in engineering.

Brooks argues that no breakthroughs in ISD productivity and quality are likely through the use of orthodox tools and techniques, because these fail to address these four essences. Instead organizations should either avoid the problems by using commercial off-the-shelf software or concentrate on producing great designs from which to develop their systems. Great designs however require a full understanding of system requirements and great designers. According to Walz et al. (1993), it is reckoned that more than half the cost of the development of complex computer-based information systems is attributable to decisions during requirements specification and design. Curtis et al. (1988) found that on large projects the three biggest problems affecting the design phase were: the thin spread of application domain

knowledge amongst developers; fluctuating and conflicting requirements and, communication and coordination breakdowns. Consequently much of the activity on projects is concerned with human skills such as learning, communication and negotiation. Individuals have to acquire and integrate knowledge from multiple domains and from different parts of the project. This learning modifies the participants understanding of the solution and can lead to changes in requirements, design and implementation throughout the project lifecycle.

We suggest that a knowledge perspective, drawing on experience and insights into knowledge management and organizational learning, provides substantial help for the both the effective management of information systems development and also the role that methodologies play in this. Cognitive theories provide an understanding of how individuals handle mental tasks and therefore how the application of ISD methods, practices and tools could be aligned to take proper account of individual factors (Robillard 1999). These factors include: the importance of learning and previous experience and how this contributes to the creation of mental schemas as models for comprehension; the problems inherent in solving ill-defined problems such as those that typically arise during design; the limitations of short-term memory and hence the importance of breaking tasks down – the famous '7±2 chunks' maxim – and how the information resources that surround the ISD team can be organized, rendered accessible and kept track of in order to facilitate good design and implementation decisions.

At the same time, individuals have to work within a larger social and organizational environment (Curtis et al. 1988). Nonaka and Takeuchi's SECI model (1995) provides an explanation of the social process by which knowledge is created and shared within an organization. Skillfully applied, Nonaka's model can inform organizational design and engender the individual learning and effectiveness described above. Central to the model is the notion that individually held tacit knowledge becomes universally held explicit knowledge through a managed process of 'socialization-externalization-combination-internalization'. This requires the careful instigation of key knowledge-transmitting relationships, the explicit expression of knowledge for group consumption and then its wider dissemination within the organization for the enrichment of other individuals' capabilities. These SECI cycles follow each other and provide an adaptable, and at least partially controllable, means to develop, transmit and apply knowledge about aspects of project that is itself changing. The SECI model emphasizes the value of integrating knowledge from various domains (both internal and external to the project) and so mitigates against the establishment of organizational 'silos' whose insularity hinders agility and directs individual energies towards the maintenance of methodological mechanisms rather than project success.

Successful systems development critically depends on users and designers learning from each other. Hence, managed, timely participation in the learning cycles, such as those described by the SECI model, is important. For example, when new members are added to teams existing members may be reluctant to use or accept the new knowledge the newcomers bring with them (Walz 1993). Conversely, how can early project participants influence later stages of a project if they are no longer actively involved? If knowledge is 'captured' once – for example user requirements gathered early in a project – and pushed into the background, it will often have less weight than the knowledge readily and currently available to the team.

With orthodox, methodologically-driven project management, a risk is that achievement of bureaucratic targets and performance measures takes precedence over the delivery of working systems and the satisfaction of end-users. In other words, the emphasis is on the process with its explicit measurement parameters and not the artifact. Knowledge management, properly applied, focuses on the more tacit and inherent qualities of the individuals, who deliver the artifact. This is not to say that performance measures or targets must be eschewed in favor of some set of loosely-defined or lofty and impractical aims. KM brings the focus back to the efficient delivery of a useful artifact and is more amenable to engaging with the communication dynamics and shorter timescales of modern ISD projects.

The literature on the learning organization (the entity) and organizational learning (the process) is a rich source of insight into the way knowledge is created and communicated. For example, Crossan's et al. (1999) model is an explanation of how the creation of new knowledge (feed forward) and the transfer of existing knowledge (feedback) occur through four processes – intuiting, interpreting, integrating and institutionalizing – at individual, group and organizational levels. Montoni et al. (2004) present an approach for acquiring and preserving ISD knowledge and making it available between organizations.

Knowledge work is not programmable and this creates challenges for managers predisposed to command-and-control management. Drucker (1999) asserts that the only way to increase the productivity of knowledge workers is to radically change the way that they are managed:

1. Non-value added activities should be completely eliminated;
2. Knowledge workers should be responsible for their own productivity;
3. Continuous innovation and learning are required, coupled with knowledge workers teaching others what they know and what they can do;
4. Quality of work is at least as important as quantity; and
5. Knowledge workers own the means of production so should be treated as an organizational asset not a cost. Managers, however, will experience tension between allowing employees autonomy and wanting to maintain control of project outcomes.

These issues have long been recognized in ISD and practice has been to grant IT professionals greater autonomy than many other business functions. Management of ISD therefore involves a relatively high degree of professional trust. Nevertheless software development almost always needs to be disciplined if it is to meet its stated objectives – i.e. allowing freedom and responsibility but within a guiding framework. Practitioners and managers alike see the benefits of methodologies in providing structure to the development process (Barry and Lang 2002) and something to manage against. Even if formalized methodologies are not used they still influence practice (Avison and Fitzgerald 2003). Methodologies also provide an organizational framework, a common language, shared paradigms and approaches to problem solving and task completion that help project participants to communicate, cooperate and learn (Wastell 1996).

In contrast, approaching ISD as a knowledge-based activity suggests that any guiding principles employed should attend to the cognitive needs of individuals, the behavioral and social realities of team based working, the desirability of individual and organizational learning and the business needs of flexibility and speed but at an appropriate level of risk.

5. Analysis of Methodologies

In order to analyze a methodology it can be considered in terms of its constituent model, techniques, tools, scope, outputs, users and practice. Also, any methodology will have an underlying intellectual framework or philosophy and will be directed towards a particular application area. It will therefore have particular management objectives and success measures.

How can the two approaches of methodologically- and knowledge and organizational learning- based views on ISD be unified in a way that meets the needs of all parties in the current environment? In the next paragraphs we propose a framework, Figure 1, that relates the two perspectives. We then highlight and discuss cross-linkages between the two.

Just as methodologies are characterized by different underlying philosophies, the knowledge management literature is broadly divided ontologically and epistemologically into objectivist and subjectivist standpoints (Ortenblad 2002). The objectivist standpoint is common in the information systems literature: knowledge is an object that can be separated from both knower and context and much valuable knowledge can be codified, stored and transmitted using information technology. This has led to the development of knowledge management tools to support ISD such as design rationale and experience factories. Kettunen (2003) follows an objectivist philosophy when he proposes methods to manage software development knowledge that involve auditing each person's knowledge and then determining with whom they should share this knowledge.

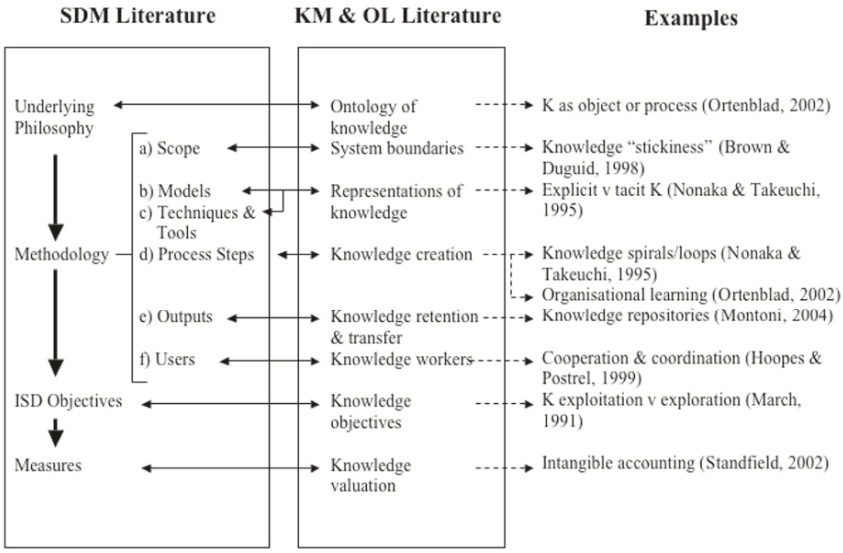


Figure 1. Mapping of KM and OL concepts onto the hierarchy of SDM elements.

The objectivist philosophy is contested by researchers who emphasize the cognitive and social aspects of knowledge and see knowledge as situated in a process of interactions (Brown and Duguid 1998). For example, the idea of communities of practice belongs to this subjectivist branch of the literature. Within these two schools there are of course numerous variations and stances.

Care must therefore be taken when combining and applying different elements of knowledge management theory in case there philosophical inconsistencies between them and the aspect of the methodology being considered. For example, during design will the emphasis be on documentation (objectivist) or workshops and walkthroughs (subjectivist)?

Different methodologies have different scopes in terms of which aspects of the systems development process that they include. This scope will determine the set and type of knowledge covered by the methodology. This can help management to judiciously determine where inter- and intra-project boundaries should be drawn; such as which user representatives should be included in the core project team. Knowledge can be "sticky" and have difficulty crossing organizational and project boundaries and so managing knowledge flow across these boundaries is an important task. There may be a need for roles responsible for translating between the different domains or to broker, mediate and coordinate the transfer of knowledge (Brown and Duguid 1998).

The way knowledge is represented is a central theme in knowledge management. The models, tools and techniques, process steps and outputs of

a methodology will determine how project knowledge is represented and therefore created, retained and transferred. Structured analysis tools, such as data flow and process flow diagrams, aim to make knowledge as explicit as possible. The story cards used in agile methodologies, on the other hand, leave much of the knowledge required to develop from them tacit. Explanations of knowledge generation and organizational learning emphasize the circulation of knowledge flows within the organization (Crossan et al. 1999; Boisot 1998; Nonaka and Takeuchi 1995), which has implications for the appropriateness of different process models as the frequency of project iterations affects how quickly a project can learn.

As for the users of methodologies, Walz (1993) found that for some projects over 75% of the time devoted to the design phase was spent in learning both the user requirements and technologies to be used. Their recommendation is that managers should increase the amount of application domain knowledge across the entire software development staff by auditing the existing knowledge of project members, allowing time for learning, and provide tools to help capture information and project experiences for later reuse.

However, learning, sharing and integration are time consuming and every member of the team does not have to know everything. Postrel (2002) suggests that given the costs of developing trans-specialist knowledge (i.e. knowledge outside of one's own specialist area) the best approach is for workers to specialize in their own knowledge domains and for management to foster "islands of shared knowledge" only where necessary. This suggests that management should be more actively involved in managing learning and the flow of knowledge than much of the ISD and product development literature proposes.

In terms of objectives, what should be the balance between knowledge exploration – the discovery and creation of new knowledge – and exploitation – the systematic and purposeful application of knowledge (March 1991)? In ISD projects there will be conflict between stakeholders because of the tensions between the working conditions conducive to thinking and creativity and the economic pressures to complete projects as quickly and cost effectively as possible. Different methodologies provide management with varying degrees of direct control over work tasks and therefore require different levels of cooperation from workers for their successful use. Although managers need to consider what individual, project and organizational learning is required to successfully develop the system that is the immediate focus of concern, it will be prudent to be mindful of what further investment should be made in learning beyond immediate project needs. This will engender competences relevant for future projects – and associated activities such as tendering.

In ISD exploration and exploitation activities often happen together but some activities, such as requirements definition and design, are more focused on exploration than others, such as coding and testing. Knowledge

exploitation in ISD is becoming progressively easier due to technology improvements in areas such as hardware, packaged software and development tools.

Similar improvements are required in knowledge exploration (Drucker 1999; Brooks 1987). The amount of knowledge exploration and therefore learning that takes place varies widely by methodology. In general iterative methodologies, such as those which incorporate a spiral process model, will encourage exploration and learning. On the other hand the more established waterfall-type models emphasize more of a knowledge exploitation approach because knowledge exploration is prescribed only in the very early phases.

However generalizations like these may be unhelpful. For example, eXtreme Programming (XP) takes the idea of multiple iterations to the extreme: ideally a project is a series of two-weekly releases. However, as Stephens and Rosenberg (2003) point out, in practice a number of XP practices are in fact anti-learning. Simple design and constant refactoring reduce the amount of reflection and thinking ahead. Pair programming may discourage an individual from working through a problem if their partner knows how to solve it. Additionally, a customer representative based on-site creates a single point of contact for all the projects external knowledge needs. XP therefore represents an extreme knowledge exploitation strategy that relies on already skilled programmers and knowledgeable customer representatives to be successful.

A final concern for managers is how performance may be measured. A knowledge based perspective emphasizes systems quality over the quantity of deliverables produced. Most of the literature on knowledge measurement and valuation is directed at the firm-level, although Standfield (2002) is creating standards for intangible accounting and management that may prove useful to ISD. The subjectivist perspective on this however is that the search for metrics is counterproductive as it attempts to reify knowledge and "such indicators do not provide any sense of an organization's stock or flow of knowledge or its contribution to decision making and organizational performance." (Fahey and Prusak 1998)

6. The Future

Fitzgerald (2000) suggests that any new methodologies should focus on simplifying and speeding up ISD by the following means:

1. Making greater use of packaged software and outsourcing, which allow systems to be developed with higher level building blocks;
2. Recognizing that most business software is algorithmically simple and therefore using methods and tools that are correspondingly straightforward;

3. Aiming for satisficing solutions that are 'good enough' for the business need rather than striving to deliver something excessively functional or sophisticated.

At the same time, methodologies should allow developers to use both SDLC (top-down) and prototyping (bottom-up) approaches to elicit requirements as needed as well as giving developers autonomy to choose their own design and implementation methods.

The above criteria, therefore, combine to change the emphasis to one that reinforces both pragmatic considerations and also that a methodology is usually there to aid rather than hinder a developer.

If future methodologies follow this line then future methodologies are likely to deal at a higher level of abstraction, specifying desired outcomes (what) rather than prescribing the exact steps to be followed (how). They should also give guidance on determining the learning needs of project staff, how requirements should be negotiated, how conflicts inherent in the creational processes can be resolved and how particular factors such as these contribute to the project's uncertainty and risk (Curtis et al. 1988).

A knowledge perspective would add to and amplify these factors by providing an assessment of knowledge needs i.e. what knowledge is available, what knowledge needs to be acquired and what knowledge will be generated; the most appropriate process model for the needs of knowledge exploration versus exploitation; the appropriate project organization and technologies for knowledge sharing, sympathetic with worker cooperation and knowledge coordination. It may be that methodologies based on the conception of ISD as an engineering discipline may become less prevalent and a new perspective and set of corresponding methods will come to the fore.

As an example of how knowledge-based thinking can contribute to a better understanding of the future of methodologies, Boisot's (1998) I-space provides a strategic model for analyzing organizations from a knowledge-based perspective. He proposes that organizational cultures can be characterized as fiefs, bureaucracies, markets and clans, which in the simplified I*-space version of his model occupy each of the four quadrants of a 2x2 matrix mapping knowledge codification against knowledge diffusion.

Boisot's bureaucracies and markets in the top half of I*-space correspond to common business usage of these terms. In bureaucracies knowledge is highly documented (codified) to allow knowledge retention and sharing, but the diffusion of this knowledge is strictly controlled within the organizational hierarchy by management. In markets knowledge is also highly codified but sharing is uncontrolled and therefore information is widely diffused. Business relationships are impersonal and competitive and coordination is through processes of mutual adjustment and self-regulation. In the bottom half of I*-space, fiefs are small organizations, such as business

start-ups, where knowledge is largely uncodified and undocumented. Relationships are face-to-face and hierarchical. Members of a fief are expected to have shared beliefs and goals and to subordinate themselves to the goals set by their leader. Finally, clans are typified by business or academic networks. The valuable knowledge in these networks is largely uncodified and passed through personal contact. A clan's goals and activities are negotiated by its members who must therefore share common values and beliefs.

Applying Boisot's I*-space to the process of information systems design provides insights into the choice of systems development methodologies. Historically for large projects, IT managers chose between in-house development using traditional, formalized methodologies and using third-party solutions either in the form of packaged software or by outsourcing software development. The newer, agile methodologies described above along with the use of open source software have been receiving increasing interest as possible software development methods. These four development approaches – formalized methodologies, packaged software and outsourcing, agile methodologies and open source software development – map onto Boisot's four organizational cultures in I*-space as shown in Figure 2.

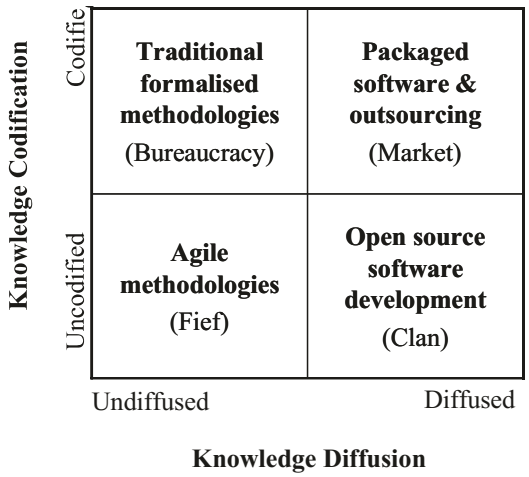


Figure 2. ISD approaches mapped onto Boisot's I*-Space.

Information technology influences organizational form. Improvements in information technology are constantly increasing the bandwidth with which information systems can process and transmit information. This increase allows organizations either to transmit more information, more quickly and more widely, increasing the diffusion of knowledge; or to reduce the amount of codification that is required prior to transmission. In consideration of these changes, Boisot hypothesizes that there will be a shift in organizational forms from bureaucracies in the top left of I*-Space to the right (greater

diffusion) and down (less codification) towards clans in the bottom right quadrant.

The emergence of agile methodologies and open source software development as viable options for software development can be attributed to the improvements in information technology hypothesized by Boisot. In the case of agile methodologies, this stems from a combination of improved modeling and programming techniques (particularly object oriented techniques) (Glass 1999) and increased machine speeds allowing fast compilation and testing. The accessibility of open source software development results from the use of the internet as a communication medium (Raymond 1997).

Organizations will continue to take positions in I*-space but this analysis suggests that there will be a move from formalized methodologies to approaches incorporating the principles of open source development (as shown by the arrow in Figure 3 above). The bottom right quadrant in I*-space (clan/open source software development) is also the culture that is suggested by much of the subjectivist literature of organizational learning (Ortenblad 2002).

7. Application to Current Practice

New knowledge-based methodologies will take time to emerge. Our research indicates that the selection of methodologies is determined as much by stakeholder preferences – which are usually towards simple to understand waterfall approaches – as the characteristics of the projects themselves. The IT departments are therefore constrained in the methodologies that they can use and see part of their role as educating the business about possible approaches.

This does not mean that knowledge management ideas cannot be usefully applied to existing practice. Recent case study research by one of the authors investigated the management of project issues during ISD using traditional methodologies. The root causes of many of the issues were found to be knowledge gaps (Hoopes and Postrel 1999): the knowledge needed to successfully perform the task existed within the project but this knowledge was not effectively used due to problems of knowledge sharing. Many knowledge sharing problems during requirements specification, systems analysis and systems design were caused by poor working relationships between the designers and other project participants.

The results of this research were used to develop a workbook – comprising methodologies to audit project performance and improve project control – to help project managers anticipate and manage the risks of knowledge gaps during the design of Information Systems. Central to this workbook was a mapping of the phases of the systems development lifecycle to Boisot's four organizational cultures in I*-space as shown in Figure 3. This mapping provided an "ideal" case for the project organization and control against which to evaluate the particular project. For example, based

on knowledge management theory, systems analysis and systems design ideally involves a fief culture involving a small, cohesive team closely supervised by the project manager. In practice the design team often involves participants from disparate organizations such as the IT department, the customer and third party suppliers and these participants may have little loyalty to the project manager or one another. The project manager therefore should be acutely aware of the risk of poor cooperation between the various participants and put in place the appropriate formal and informal control modes to manage their work, as well as contingencies to deal with any residual risks.

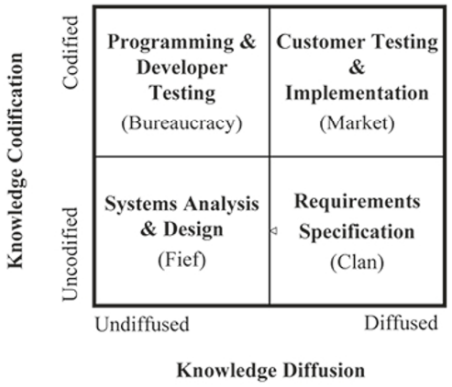


Figure 3. SDLC phases mapped onto Boisot's I*-Space.

8. Next Steps and Conclusion

This paper has presented an overview of the history, theory and state of practice of systems development methodologies. It has also presented some ideas from the knowledge management literature and proposed that these should be incorporated into the future development of methodologies and so provide greater structure for the coordination of activities within ISD projects. Given the different viewpoints about the nature of knowledge and the role of management, it is certain that – as with existing methodologies – knowledge-based methodologies will be developed that reflect both different philosophies, types of projects and stages of information systems development.

We are pursuing two streams of work. The first involves gathering empirical evidence for the value of the knowledge perspective in practice. Practice has often preceded theory in the field (Fitzgerald 2000) and research is being directed at examining whether knowledge management and organizational learning ideas are in fact influencing ISD practice. The second stream is developing knowledge-based methodologies following the

guidelines presented in this paper with the intention of using them in action research.

References

Abrahamsson, P: 2002, *Agile Software Development Methods. Review and Analysis,* VTT Electronics, Oulu, Finland.

Avison, DE and Fitzgerald, G: 2003, Where now for development methodologies?, *Communications of the ACM* **46**(1): 79-82.

Avison, DE and Wood-Harper, AT: 1991, Information systems development research: An exploration of ideas in practice, *The Computer Journal* **34**(2): 98-112.

Barry, C and Lang, M: 2002, A comparison of 'traditional' and multimedia information systems development practices, *Information and Software Technology* **45**(4): 217-227.

Boehm, B and Turner, R: 2003, *Balancing Agility and Discipline: A Guide for the Perplexed,* Addison-Wesley, Boston.

Boisot, MH: 1998, *Knowledge Assets: Securing Competitive Advantage in the Information Economy,* Oxford University Press, Oxford.

Brooks, FP: 1987, No silver bullet: Essence and accidents of software engineering, *Computer* **20**(4): 10-19.

Brown, JS and Duguid, P: 1998, Organizing knowledge, *California Management Review* **40**(3): 90-111.

Crossan, MM, Lane, HW, White, RE: 1999, An organizational learning framework: From intuition to institution, *Academy Of Management Review* **24**(3): 522-537.

Curtis, B, Krasner, H and Iscoe, N: 1988, A field study of the software design process for large systems, *Communications of the ACM* **31**(11): 1268-1287.

Drucker, PF: 1999, Knowledge-worker productivity: The biggest challenge, *California Management Review* **41**(2): 79-94.

Fahey, L and Prusak, L: 1998, The eleven deadliest sins of knowledge management, *California Management Review* **40**(3): 265 - 276.

Fitzgerald, B: 2000, Systems development methodologies: The problem of tenses, *Information Technology and People* **13**(3): 174-185.

Glass, R: 1999, The realities of software technology payoffs, *Communications of the ACM* **42**(2): 74-79.

Hoopes, DG and Postrel, S: 1999, Shared knowledge, "glitches," and product development performance, *Strategic Management Journal* **20**(9): 837 - 865.

Kettunen, P: 2003, Managing embedded software project team knowledge, *IEE Proceedings - Software* **150**(6): 359-366.

March, JG: 1991, Exploration and exploitation in organizational learning, *Organization Science* **2**(1): 71-87.

Montoni, M, Miranda, R, Rocha, A and Travassos, G (eds): 2004, *Knowledge Acquisition and Communities of Practice: An Approach to Convert Individual Knowledge into Multi-Organizational Knowledge*, Lecture Notes in Computer Science 3096, Springer.

Nonaka, I and Takeuchi, H: 1995, *The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation,* Oxford University Press, New York.

Ortenblad, A: 2002, Organizational learning: A radical perspective, *International Journal of Management Reviews* **4**(1): 87-100.

Postrel, S: 2002, Islands of shared knowledge: Specialization and mutual understanding in problem-solving teams, *Organization Science* **13**(3): 303-320.

Raymond, ES: 1997, *The Cathedral and the Bazaar*, Available Online, http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

Riemenschneider, CK: 2002, Explaining software developer acceptance of methodologies: A comparison of five theoretical models, *IEEE Transactions on Software Engineering* **28**(12): 1135-1145.

Robillard, PN: 1999, The role of knowledge in software development, *Communications of the ACM* **42**(1): 87-92.

Standfield, K: 2002, *Intangible Management: Tools for Solving the Accounting and Management Crisis,* Academic Press, San Diego.

Stephens, M and Rosenberg, D: 2003, *Extreme Programming Refactored: The Case Against XP,* Apress, Berkeley, CA.

Walz, DB: 1993, Inside a software design team: Knowledge acquisition, sharing and integration, *Communications of the ACM* **36**(10): 63-77.

Wastell, DG: 1996, The fetish of technique: Methodology as a social defence, *Information Systems Journal* **6**(1): 25-40.

ANALOGICAL MATCHING USING DEVICE-CENTRIC AND ENVIRONMENT-CENTRIC REPRESENTATIONS OF FUNCTION

GREG P MILETTE AND DAVID C BROWN
Worcester Polytechnic Institute, USA

Abstract. This research experiments with representations of function using analogical matching, trying to determine the benefits of using environment-centric (EC) vs. device-centric (DC) representations. We use the Structure Mapping Engine for matching, and seek to show the effect on quality and quantity of analogical matches when the representation is varied.

1. Introduction

Designing something is challenging, so providing computational help is important. Software systems can help the designer, or might replace the designer in some situations (Brown 1992). For computers to design devices we have to describe them in some way using a knowledge representation.

This research experiments with two different knowledge representations: both based on the Structure-Behavior-Function model for describing devices (Chandrasekaran and Josephson 2000). That work describes two different ways to represent the function of devices, Device-Centric (DC) and Environment-Centric (EC), both described below. Each may be advantageous in certain situations, but there appears to be no research showing what the effects are of using DC vs. EC representations of function.

This research seeks to perform experiments that will explore what the effects are with certain design tasks. The knowledge representation will be used in experiments with some automated reasoning, producing results that we can measure.

Motivated by exploring computational support for creativity (Boden 1994), we target analogical reasoning. Analogy is often cited as a key ingredient of creativity (Goel 1997; Gentner et al. 2001). In addition, functional reasoning is also at the right level of abstraction to support creativity (Umeda and Tomiyama 1997).

Analogical reasoning involves expressing what the current situation is, looking for past situations that might apply (matching), and finally applying

J.S. Gero (ed.), Design Computing and Cognition '06, 183–202.
© 2006 *Springer. Printed in the Netherlands.*

them to the current situation (transfer). A full study would require a system that performs all the steps in analogical reasoning, but for this research we take the first step and focus only on the matching phase. We use an algorithm called SME (Falkenhainer et al. 1989). SME was chosen because it is well tested in much research, it is claimed to have psychological backing, the software is available, and because it is suited for the problem.

Using SME we can take a pair of devices represented with a particular knowledge representation and produce a list of possible matches between them with associated weights. We measure the quantity and quality of the matches in order to measure the effect of DC vs. EC representations.

We are also interested to see whether novel matches are produced: i.e., whether DC vs. EC representations might have any effect on novelty, a key aspect of creativity (Besemer and Treffinger 1982). Therefore we consult a group of humans to get their judgment.

We have performed a set of experiments that that indicate where the results are coming from: i.e., the credit assignment problem. The issue is whether DC vs. EC representations, or the representation used (level of detail; ontology) should be given credit (Kitamura et al. 2004).

The *hypothesis* of this research is that representations with EC information will produce a greater number of analogical matches, and that these matches will be of lower strength than matches made using representations that only contain DC information. We hypothesize that creating a representation with *both* DC and EC information should produce even more matches than either DC or EC alone, and that these matches should have higher weights.

We show through experimentation with SME that EC produces more matches than DC, DC produces higher quality matches than EC, and, in contrast to our hypothesis, the combined representation produces comparatively fewer matches and more lower quality matches than EC alone.

In addition, from limited experiments with humans we show that they tend to rate low weighted matches as being more novel than high weighted matches and rate DC matches as being more novel than EC matches.

2. Design Research

A lot of research has been done on functional representation and reasoning (Umeda and Tomiyama 1997; Stone and Wood 1999; Pahl and Beitz 2003; Chandrasekaran and Josephson 2000; Stone and Chakrabarti 2005.

A full description of a device D's intended function would link it, via relationships and behaviors, to "purpose" Brown and Blessing 2005; Rosenman and Gero 1998). There is little work addressing the effect of a device on the environment. Prabhakar and Goel (1996) distinguish the

"external environment" of D, from the "outer environment" of D: i.e., those entities in the environment that directly interact with D.

Our work is most influenced by Chandrasekaran and Josephson's framework (2000). They consider that a device (D) is used by being placed in an environment (E). The causal interactions that result from this "mode of deployment" occur due to a pattern of relationships over time between D and E. If the pattern of behaviors arising from the interactions is *desired* then D performs a function in E. Behaviors are seen as values of, or relations between, state variables, or properties, of an object, considered over time.

If the desired behaviors are expressed in terms of D only, then they consider it to be a Device-Centric (DC) description of D's function. An Environment-Centric (EC) description only uses elements in E: such a description might be presented in the early stages of a design task.

Our representations are a strict DC version, and another that includes all entities that interact, from both D *and* the outer environment. We refer to this, for contrast, as EC, but from Chandrasekaran and Josephson's definitions it should more properly be called "mixed".

2.1. USING THE FUNCTIONAL BASIS

The terms used to describe the function of different devices must be consistent and at the same level of abstraction so that device descriptions are comparable. This will reduce the variation and noise in results. For example, using more abstract terms for one device may cause SME to generate more matches, making strong conclusions harder to make, while inconsistent terms may cause fewer matches, with similar consequences.

This research uses a set of domain specific terms called the "functional basis" (Stone and Wood 1991). The functional basis provides a set of domain-dependent terms for flows and functions. The representations in this research use flows in the same way the functional basis does. The functional basis represents flows of material, energy or signal that transfer from one device to the next. The basic functions available include import, export, transmit, couple, display, rotate, and change. Our representation uses the basic functions from the functional basis work as a way of describing device behaviors.

3. Knowledge Representation

There are several goals for the knowledge representation (KR):

1. It can represent DC and EC functions;
2. It can represent devices at different levels of detail;
3. DC and EC parts can be combined to form a combined representation. We refer to this representation as BOTH (see example in Section 3.3).

The KR must be descriptive enough to describe functions and must allow for different experiments. These experiments (Section 6) require the ability to represent devices at different levels of detail, and also to use the DC only, EC only, or BOTH versions of the each device's representation.

3.1. DESIGN DECISIONS

We consider a function to be a set of desired behaviors. Rather than including all of the constructs from Chandrasekaran and Josephson's work, such as mode of deployment, this research represents only behaviors and functions, leaving further exploration of Chandrasekaran and Josephson's concepts to future work.

The KR is somewhat independent from SME vocabulary, but is still easily translatable into proper input for SME. This decouples the KR from the particular intricacies of the matching algorithm implementation used.

3.2. OBJECTS IN THE REPRESENTATION

There are five main concepts in the KR: devices, functions, behaviors, relations, and flows. To completely specify a device using the KR, one must provide a library of relations and flows, a set of behaviors and a set of functions that group the behaviors.

A device has a set of functions that are either DC or EC. Each function consists of a set of behaviors. Since a device may have multiple functions, some of a device's behaviors may be mentioned in more than one function.

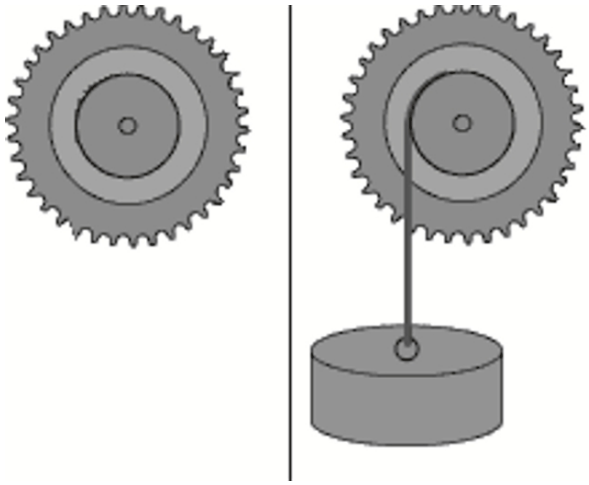
Devices are physical objects in the world and their behaviors describe how they interact. Behaviors are instantiations of relations. The relations (e.g., import) provide constructs that are filled in with domain specific elements, such as flows or other devices, in order to specify a behavior. For example "import <flow> <device>" is an example of a relation with two arguments. Instances are import torque gear and import force drum.

Flows are the material, energy or signals involved in a particular behavior. For example, a behavior change force surface describes how the flow "force" interacts with the device "surface".

The environment for a particular device is an outer environment defined by a set of external objects that interact with the device. It is *not* the entire external environment. The representation does not have an explicit representation of the environment. Instead it describes the environment using behaviors. For example, the behavior transmit torque *minutegear* references *minutegear*, which is part of the environment. Also, the representation can have behaviors that do not refer to the environment at all. To distinguish objects which are part of the environment from the device we mark objects in the environment by underlining them.

3.3. USING THE KNOWLEDGE REPRESENTATION

The objects described in Section 3.2 can be used to satisfy the goals we had for the knowledge representation. This section provides examples of devices represented with high and low detail. This section also provides examples of DC and EC behaviors and functions.



(a) (b)

Figure 1. (a) A gear (b) a gear and a weight. Other devices that interact with these two, earth and gear2, are not shown.

The KR can be used to represent DC behaviors and functions for the gear pictured in Figure 1(a)s. The relation "import <flow> <device>" is used to define the behavior:

import force gear (b1)

The relation "export <flow> <device>" is used to describe the result of behavior b1:

export force gear (b2)

The two behaviors combine to form a single DC function.

b1, b2(dc1)

To represent EC behaviors and functions, the representation needs to introduce another device to interact with the gear because EC behaviors need to mention something in the environment.

For the situation with a weight and two gears, partially represented in Figure 1b, two EC behaviors are available for the gear:
transmit force from *weight* to gear (b3)
transmit force from gear to *gear2* (b4)

The environment of the gear consists of *weight* and *gear2*. The behaviors b3 and b4 combine to form an EC function:

b3, b4 (ec1)

The weight in the mechanism can also be represented with two behaviors:

transmit force from *earth* to weight (b5)
transmit force from weight to *gear* (b6)

The environment of the weight consists of *earth* and *gear*. The behaviors b5 and b6 combine to form an EC function for the weight:

b5, b6 (ec2)

When representing with low detail, the representation focuses on a particular device. The device has no internal components and the behaviors for the device either refer to the device itself or to objects in the environment.

For a high detail representation, the KR needs to combine low detail descriptions together. The representation does this by combining the behaviors and functions from each low detail device. Using the gear and weight example from figure 1b the high detail EC function would contain four behaviors instead of two and only one function. The EC function would be:

b3, b4, b5, b6 (ec3)

This KR can be used to create a BOTH representation by concatenating the EC and DC version of each device representation. Thus, the BOTH representation for the gear consists of the functions dc1 and ec1 as well as the behaviors b1, b2, b3, and b4. Note that as the DC and EC representations use different relations there is no overlap when constructing the BOTH representation.

4. SME

This section briefly describes how SME works and what features are relevant for this research (Falkenhainer et al. 1989). The code used in this research is available online (Falkenhainer 2005).

The SME algorithm takes two devices, called the source and target, and maps knowledge from the source into the target. The first step of the algorithm is to create a set of match hypotheses. A match hypothesis represents a possible mapping between a part of the source and the target.

SME uses match rules to calculate positive and negative evidence for each match. SME combines different amounts of evidence together,

favoring matches between parts of the device representation that have similar relation names.

For example, given relations r1 and r2:

transmit torque inputgear secondgear (r1)
transmit signal switch div10 (r2)

SME produces four match hypotheses. Three of the match hypotheses are torque to *signal*, *inputgear* to *switch*, and *secondgear* to *div10*. Each has a positive evidence value of 0.6320. SME also matches both "transmit" relations and gives them 0.7900 positive evidence. These matches are between flows, between devices, and between relations. All of these matches are dependent on the shared relation name "transmit".

The match rules also propagate evidence from higher matches down to lower matches. This gives additional evidence to matches that are part of a higher order relation match. SME does this because of the Systematicity Principle which states that more connected knowledge is preferred over independent facts (Falkenhainer et al. 1989).

SME can produce negative evidence when the relation type matches, but the elements in the relation do not match.

The rest of the SME algorithm is involved in creating maximally consistent sets of match hypotheses. These sets are called "gmaps". The sum of all the positive evidence values in the match hypotheses of a gmap becomes the weight of the gmap. Comparing a source device to a target device may produce one or more of these gmaps, each with an associated weight. Also, SME combines multiple smaller gmaps to produce bigger, maximally consistent, gmaps.

4.1. SME PROPERTIES RELEVANT TO THIS RESEARCH

Since this research is comparing two kinds of KR, there are some properties of SME that are relevant for the determining reasons why one KR produces different results than another:

- More information in a particular representation should allow for matches of higher weight. This is because longer representations can produce more match hypotheses and thus have higher weighted gmaps.
- Making longer representations may not produce a greater number of gmaps because gmaps can be combined together during the creation of maximally consistent gmaps

Our experiments use these properties to explain why results using the DC and EC representations differ.

5. Test Examples

The requirements for the test examples to be used are that they: must have varied levels of detail; must include both DC and EC representations; should be similar enough to allow analogical matches; should allow for novel matches; must be a large enough sample so that general conclusions can be reached; and must be capable of being understood by humans.

The test examples used in this research are a set of clocks that are decomposed into components and subcomponents. By combining different subcomponents together, the level of detail can be adjusted. Because the clocks share components, there are obvious analogical matches that SME can make, providing good contrast for results that people may consider novel. The test examples represent 21 individual subcomponents, which can be grouped into 8 larger components.

5.1. THE CLOCK TEST EXAMPLES

We use two kinds of clocks: a digital clock, such as a bedroom alarm clock, and a pendulum clock, such as a grandfather clock. Each clock has a different way to achieve the functions of setting and displaying the time.

Each clock works differently, but they share common components and common functions. These components are the powerprovider, which provides some kind of energy into the clock, the timebase, which converts the energy into a periodic signal, a gear, which converts the signal into a once-per-second or once-per-minute signal, and a face which displays the time.

We used articles by Brain (2005a; 2005b) as sources of information about clocks. When using a clock a human needs to observe the time and be able to set the time. Figure 2 shows a conceptual diagram of these components and how they interact. Arrows indicate the direction of flow in the clock. For example, the powerprovider transfers energy to the timebase. The human interacts with the clock by resetting it or by receiving a visual signal.

Figure 3 shows a schematic for a pendulum clock. The schematic labels all the pendulum clock's components. Figure 4 shows how these subcomponents get grouped into components. For example, the secondhand and minutehand are subcomponents of face. The schematic for the pendulum clock is shown in Figure 5.

The pendulum clock works primarily with gears while the digital clock uses many divide-by-x counters. The hierarchy for the digital clock includes subcomponents such as a divide-by-10 counter, which is part of the digital gear, and a plug, which is part of the digital power provider.

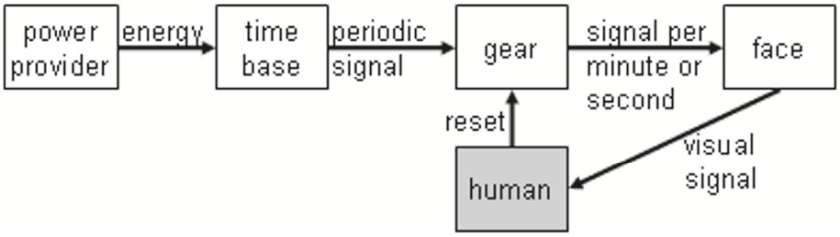


Figure 2. Generic model of a clock: components and how they interact with each other and with a human.

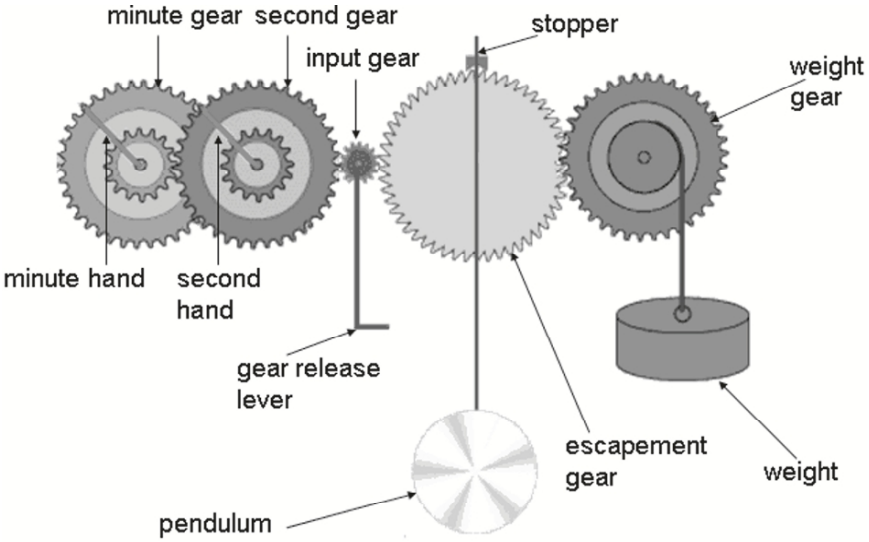


Figure 3. Schematic for an idealized pendulum clock showing all its components. Diagram based on (Brain 2005b).

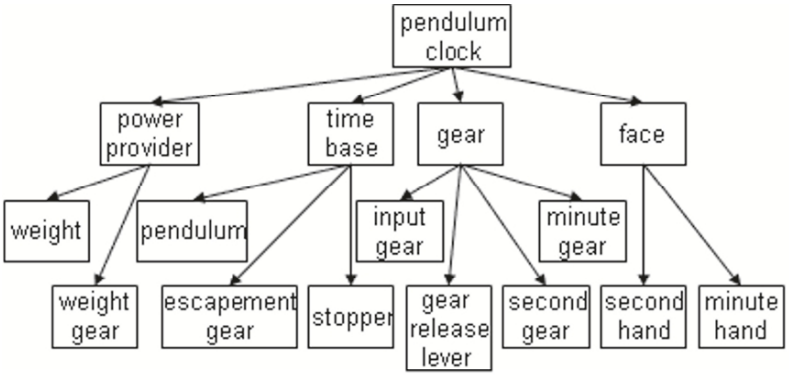


Figure 4. Hierarchy for the pendulum clock. Boxes show the devices; arrows represent a component-subcomponent grouping.

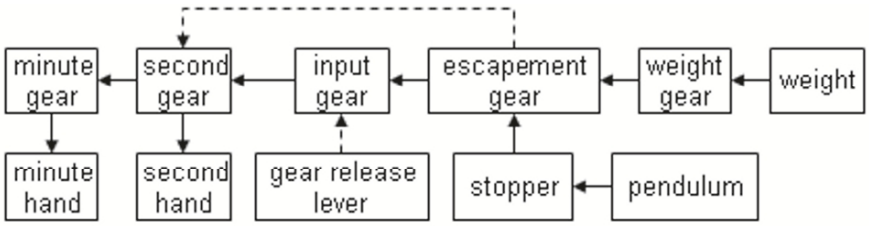


Figure 5. Schematic for pendulum clock. Boxes represent subcomponents; solid arrows represent flow; the dotted line represents flow when the gear release lever is pressed.

Thus, the test examples are made up of two different clocks that can be represented at two levels of detail. The low detail representations are the subcomponents of the clocks such as *secondgear* or *plug*. The high detail representations include clock components such as digital *powerprovider*.

6. Computational Experiment

SME produces a list of gmaps for each match, each with an associated weight. The goal of the computational experiment is to analyze these lists of gmaps and explain how they are affected by different representation types. Overall, the experiment demonstrates the following effects:

- EC has lower weighted matches than DC;
- EC generates more matches than DC;
- EC matches have higher variance than DC;
- BOTH matches are fewer in number and have lower weights than DC or EC alone.

The experiment and analysis must be able to measure these effects, explain them, and show that they are robust. The experiment measures the gmap weights, the gmap weight variance, and the number of gmaps generated. To make fair comparisons between the datasets the gmap weights and number of gmaps are normalized.

The experimental results can be influenced by several factors including the representation length, the representation complexity, and the number of devices mentioned. Each experiment is run on low and high detail test examples in order to show that any observed effects remain the same even when the level of detail is varied.

6.1. EXPERIMENTAL RUNS

The experiment uses the factorial experiment design shown in Table 1. Overall, there are 6 different device sets that the experiment uses. There are

three versions of device representations, EC, DC, and BOTH. Each version is categorized into low detail and high detail. Each combination makes up an experiment test set. There are 21 low detail and 8 high detail devices. The devices are the 21 subcomponents and 8 components of the clocks described in Section 5.

TABLE 1. Factorial experiment design showing the 6 different device sets.

	Low detail	**High detail**
EC		
DC		
BOTH		

An experiment test run consists of analyzing pairs of devices from a particular test set. SME compares each device in the test set to the other devices in the test set. The experiment disregards comparisons between the same device. This results in n^2–n comparisons where n is the number of devices in the test set. For example, the low detail test set has 420 matches in it.

6.2. EXPERIMENTAL FACTORS

This experiment needs to show how the gmap weight, gmap weight variance, and number of gmaps differ for the EC, DC, and BOTH datasets. This is complicated by the fact that several factors can affect these statistics.

The representation length is the sum of the number of functions and behaviors in the source representation. We find that for most of the data, the representation length and the number of gmaps are positively correlated ($p<0.05$). This means that as the representation length increases more gmaps get generated. Our normalization procedure decreases this correlation.

The representation complexity is the sum of the number of behaviors in each function and the number of arguments in each behavior divided by the representation length. For example, the DC version of the gear from Section 3.3, with behaviors b1, b2 and function dc1, has a representation complexity of 2. This measure of complexity is similar to the one used in (Balazs 1999).

In our data, on average, EC representations have the highest amounts of complexity. This is because DC representations only mention the device, and EC representations mention both the device and the environment.

6.3. NORMALIZED GMAP WEIGHT AND VARIANCE

The experiment needs to compare the magnitude and variance of the weights between the datasets. The factors described in Section 6.2 imply that the gmap weights cannot be compared directly unless some aspects of the representation are taken into account.

Therefore, we use a normalization strategy in order to make a fair comparison between the representations. The normalization formula first computes the value, MAXVAL, which is equal to the highest weighted gmap SME produces when the device is compared to itself. Then the weight of each gmap made with that device is divided by MAXVAL to obtain a new normalized weight.

This strategy should adjust the magnitudes of the gmap weights to account for both the representation length and complexity. It also gives the measurement more meaning. Instead of measuring its overall strength, this normalized weight measures the relative amount of a device's representation that is matched by the target device. Thus, the higher the normalized weight, the more of the target device fits with the source device.

Each time SME generates a match it outputs a list of gmaps, each of which has an associated weight. Since our comparisons are done on a per match basis and not on a per gmap basis we need to aggregate the gmap weights for each match and then use the aggregated result for our analysis. Thus, for each match, we compute the average, standard deviation, and highest of its gmap weights. Then, for all matches we compute additional statistics to create results such as "average of average gmap weights" or "average standard deviation of gmaps."

6.4. NORMALIZED NUMBER OF GMAPS

The number of gmaps is positively correlated with the representation length. In order to account for this influence and to compare the different datasets, we normalize the data by the representation length. Unlike the gmap weight measure, we could not use the number of gmaps generated when a device is compared to itself because it was not close to an upper bound on the number of gmaps.

The formula for computing the normalized number of gmaps is the number of gmaps divided by the representation length. For example, if a match has a representation length of 5 and generates 10 gmaps, then the normalized number of gmaps would be 2.

7. Human Experiment

In the computational experiments we present SME with representations of two devices, and it outputs a list of potential matches between portions of each representation. For example, based on these lower level matches, SME might suggest that a pen is like a hammer.

As our hypothesis concerns the possible benefits of different styles of device representation, the representation is varied throughout the experiments, and the resulting matches are measured and evaluated.

We are interested in performing the human experiment for two reasons. First, we would like to determine whether or not the matches proposed by SME are "novel": e.g., a pen is like a sponge. We hypothesize that one form of device representation is less likely to produce novel matches.

Second, we would like to investigate how the match weights generated by respondents correlate with SME match weights. There are two ways the SME results can correlate.

First, the human and SME results could place the same relative weights on certain device matches. For example, both the human and SME could think that the pen is more like a hammer than it is like a sponge.

Second, the human results can lend support to the DC or EC representation if the reasons the humans are using match with the representation that SME uses, and if the human's and SME's match weights correlate. We might get this result if the human thought the pen was most like a sponge because they both interact with liquid and if SME marked them as most similar because the pen and sponge both interact with a human's hand. Though the reasons are not exactly similar, they both involve EC reasoning: i.e., about how the device interacts with the environment.

To gather this information from human respondents, we use two techniques: repertory grids and a questionnaire (Hart 1986). The respondents are a volunteer group of engineers.

The repertory grid technique provides several benefits:

- It is a proven technique that allows respondents to give information about the similarity of different devices in a group. The result of collecting the grid information is a "percent similar" measure describing the human's evaluation of device similarity. After normalization, it can be compared to SME output which also reports how similar devices are.
- As part of the grid creation process, respondents give reasons why they differentiated one device from another. This information can be classified as DC or EC, lending support to that approach. It can also be compared directly to the lower level matches in the computer results.
- A good computer tool is available that makes collection of repertory grids relatively easy (Shaw and Gaines 2005).

A questionnaire is used to determine novelty. It asks the respondents to evaluate how novel they think the computer's analogical matches are and to disregard other observations such as correctness when they make their evaluation. The respondents indicate low, medium, or high novelty. The respondents are asked about results from various computer experiments produced using different representations.

The experimental procedure is to first collect a repertory grid and then have the respondent fill out a questionnaire: both about the same devices. Collecting the repertory grid first is important, as it allows the respondents to determine for themselves how the devices relate to each other. Thus, when

they fill out the questionnaire, they will be able to compare the computer's answers to their own and be better able to judge the novelty of them.

A subset of the clock examples from the computational experiment is used for the human experiment. Preliminary experiments with simpler examples, such as pens and sponges, indicated that the respondents were using reasons that we could classify as DC or EC. However, many of their reasons were focused on surface features. The clock examples subsequently adopted have similar functions, but very different surface features. Therefore, the respondents tend to focus their attention on the function of the clock components, which is what we want.

Since understanding clocks takes time, the respondents were given articles to read before the experiment. These articles were the same ones that we used to create the representations for the computational experiment, taken from *How Digital Clocks Work* (Brain 2005a) and *How Pendulum Clocks Work* (Brain 2005b). The respondents were engineers, and had little trouble understanding the examples, given the documentation.

8. Results

8.1. COMPUTATIONAL RESULTS

Tables 2 to 5 show the averages of the computational results for the various measurements in the experiment. Higher values indicate a stronger match.

TABLE 2. Average of average normalised group.

	Low detail	**High detail**
EC	0.5543	0.4705
DC	0.6907	0.5580
BOTH	0.4390	0.3935

TABLE 3. Average highest normalized gmap weight per match.

	Low detail	**High detail**
EC	0.7629	0.6460
DC	0.6907	0.6086
BOTH	0.7081	0.6193

8.1.1. DC and EC Comparison

Our hypothesis concerning gmap weights was that the DC weights would be higher than EC weights. This is true for average gmap weight, but not true for highest gmap weight. The difference for low detail result, Table 2, is statistically significant ($p<0.05$), while the difference for high detail is not.

TABLE 4. Average standard deviations of normalized gmap weights per match.

	Low detail	**High detail**
EC	0.1796	0.1212
DC	0.0	0.0435
BOTH	0.2512	0.1275

TABLE 5. Normalized number of gmaps per match.

	Low detail	**High detail**
EC	0.9421	2.5664
DC	0.2952	1.2389
BOTH	0.4883	1.9624

This can be explained by the standard deviations in Table 4: it shows that the standard deviation for EC is higher than it is for DC. The standard deviations are statistically different ($p<0.05$). Although the EC representation might have a few gmaps with higher weights, it has other lower weighted gmaps that decrease the match's average gmap weight. Thus, in the experiments DC representations produced a few high weighted matches that have similar weights while, the EC representations produced matches that have a wider variety of weights. This resulted in lower average gmap weights and higher highest gmap weights for the EC representation.

Another one of our hypotheses was that EC would produce more matches than DC. The data, shown in Table 5, shows that EC produces at least twice as many gmaps as DC. This result is statistically significant ($p<0.05$).

8.1.2. BOTH Dataset

Our final hypothesis is that the matches from the BOTH dataset will have more matches of higher weights than the DC or EC datasets. This makes sense because the more information the representation has, the more it should be able to match.

Our results show the hypothesis is correct for absolute gmap weights, but not for the normalized weights. The normalized weights measure how much of the representation was matched. This result means that a large portion of the BOTH representation is left unused in each gmap.

We observed that gmap weights from the BOTH dataset have a lower highest gmap weight than the ones from the EC dataset and only a slightly higher highest gmap weight than the ones from DC dataset. The EC dataset has statistically different highest gmap weights and the DC dataset does not have statistically different highest gmap weights.

We also observed that the average gmap weight for the BOTH dataset was lower than it was for the DC and EC datasets. This effect is partly caused by the fact that BOTH has a higher standard deviation than DC.

However, this does not explain the difference the BOTH dataset has with the EC dataset, because they have about the same standard deviation. A statistical test did not reject the possibility that the standard deviations are similar.

One explanation for this is that when DC and EC information are together the DC information is preventing the matches that would have been generated if only the EC information was present. It could be that with the BOTH representation it is harder to make globally consistent gmaps, as there is so much data with which to be globally consistent. Because the normalization discounts for not having large matches, the match weights are lower.

Another observation about the BOTH dataset is that its highest gmap weight and number of gmap measures are in between DC and EC measures. It seems that adding EC information to the DC information improved the highest gmap weight and number of gmaps by only 24% to 55% of what would have been gained by using the EC information only. Investigating this further, we found that the number of gmap weights from the BOTH dataset is not statistically different from a dataset made by averaging the number of gmap weights from the DC and EC datasets. The average number of gmaps for the averaged DC/EC dataset is 1.8245, which is close to the value of 1.9624 for BOTH.

8.1.3. Robustness to level of detail

With a few exceptions, these observations are robust to changes in the detail of the representation. The data shows that the same trends occur in the low detail as in the high detail data. The observations that are different are caused by special properties of the low detail data. One difference is that the DC representation seems to be less effective in low detail devices than in high. The low detail DC representations produced one gmap at most for any matches. The high detail representation, however, did not have this problem. We conclude that the low detail representation is too small for our DC representation.

8.2. HUMAN RESULT ANALYSIS

We collected data from 10 respondents: a repertory grid and a questionnaire for each respondent. This section offers the results from this limited survey.

8.2.1. Repertory Grid

We use the “percent similar” measure generated by the repertory grids collected from the respondents, and compare that measure to the normalized highest gmap weight generated by SME. Each repertory grid was made between 6 devices, making 36 possible evaluations between devices.

Although percent similar can range between 0 and 1, it should not be directly compared to SME data since the repertory grid collection technique asks for clarification when similarity levels are above a certain percent. This makes the percentages artificially low. Therefore, we compute *match rankings* based on the percent similar measures generated by SME and the repertory grid.

We looked for correlations between the DC and EC data sets and each of the 10 individual respondents. We use the Spearman rank order test to detect correlation between the datasets and the Wilcoxon signed rank for testing that the medians of the differences between the datasets are different.

The tests show no significant correlations between the DC and EC datasets and the respondents' answers ($p>0.23$). The data also shows that the datasets and the respondents' answers are significantly different ($p<0.1$).

The repertory grid is also used to try to determine whether the respondents' reasons given in the repertory grid correlate with the SME datasets. We classified each of the respondent's constructs used in their repertory grid as DC, EC, or neither.

One of the respondents with the most EC constructs had the strongest correlation with the EC dataset. The respondent with the most DC constructs was slightly more correlated with the EC dataset. The respondent that was most correlated with the DC dataset had 4 EC constructs and 2 DC constructs. Our analysis also shows that sometimes, the classification of the respondents' constructs predicts which dataset they will be more correlated with. This occurred in the data from 5 of the 10 respondents.

8.2.2. Questionnaire

The Questionnaire consisted of 8 questions about novelty. Four of the questions were from DC matches and the other 4 were from EC matches. The questions spanned matches that SME gave high and low match weights to: i.e., high m-weight questions and low m-weight questions. Overall, the respondents marked 21 with high novelty, 30 with medium, and 29 with low.

First, we expected that EC matches would be more novel because EC can make a wider variety of matches. However, we discovered that the respondents considered the DC matches slightly more novel. Twelve of the 21 high novelty scores were for DC.

Second, we expected that the lower the SME match weight, the more novel the respondents would rate the match. Since a lower weight means that the match was not a very strong match, we expected that lower weighted matches would seem more original to the respondents.

The respondents' data shows this effect, Table 6. There were 5 high m-weight questions and 3 low m-weight questions. Nine of 21 high novelty ratings were given to the low m-weight questions for an average of 3 high

novelty ratings per low m-weight question and 2.4 high novelty ratings per high m-weight question.

TABLE 6. Average number of novelty ratings per question class.

	High novelty	**Medium novelty**	**Low novelty**
High m-weight q's	2.4	4.4	3.2
Low m-weight q's	3	2.6	4.3

9. Discussion

The purpose of this research is to explore the differences between DC and EC representations of function. To do this we created a KR and represented a set of clock test examples. We performed a computational experiment with SME and performed an informal human experiment. From these we have discovered some properties of DC and EC representations that may be useful for computer-based design systems and the designers who use them.

First, our experiment shows how a designer might use a knowledge representation more effectively to generate novel matches. Our human experiment shows that the respondents determined low weighted matches to be more novel than high weighted matches. Our computational experiment shows that EC representations produce the most matches, some of which are low weighted. This suggests that to find novel matches a designer should prefer representations that are EC.

In addition to this, our computational experiments show that to make novel matches, the designer should not mix strictly DC representations and EC representations for several reasons. First, the experiments show that although the matches from the BOTH representation were as varied as EC, there were not as many. The experiments also show that adding extra DC information to EC representations causes them to perform worse than the EC representation alone.

Another way to produce novel matches is to use a DC representation alone. Our human experiment shows that the respondents rated matches that came from DC representations as being more slightly novel than matches from EC representations.

Unfortunately, our results are inconclusive about whether DC or EC representations are more useful for generating novel matches. On one hand, the low weighted matches that EC representations create can generate novel results. On the other hand, DC representations, which produce few low weighted matches, can also produce novel matches. Thus, more work needs to be done in order to determine which has a greater effect on producing novel matches.

Another result is that DC representations are useful for when the designer is looking for a few strong matches. By using a DC representation, the designer can expect to get fewer matches to sort though, and to find matches that are more relevant to their work. Chandrasekaran and Josephson (2000) say that it may be beneficial for designers to switch focuses from EC to DC at a certain point in the design process. This research suggests that this decision point may be when the designer wants the design system to produce fewer, more focused matches.

References

Balazs, ME: 1999, *Design Simplification by Analogical Reasoning*. Ph.D. thesis, Worcester Polytechnic Institute, Computer Science Department, Worcester, MA, USA.

Besemer, SP and Treffinger, DJ: 1999, Analysis of creative products: Review and synthesis, *in* GJ Puccio and MC Murdock (eds) *Creativity Assessment: Readings and Resources*, Creative Education Foundation Press, Buffalo, NY. pp. 59-76.

Brain, M: 2005a, *How Digital Clocks Work*, http://home.howstuffworks.com/digital-clock.htm

Brain, M: 2005b, *How Pendulum Clocks Work*, http://home.howstuffworks.com/clock.htm

Boden, MA: 1994, What is creativity?, *in* MA Boden (ed), *Dimensions of Creativity*, MIT Press, Cambridge, MA, pp. 75-117.

Brown, DC: 1992, Design, *Encyclopedia of Artificial Intelligence*, John Wiley, 2nd ed, pp. 331-339.

Brown, DC and Blessing, L: 2005, The relationship between function and affordance, *Proc. ASME Design Theory and Methodology Conference*, DETC2005-85017.

Chandrasekaran, B and Josephson, JR: 2000, Function in device representation, *Engineering with Computers* **16**: 162-177.

Falkenhainer, B: 2005, *Structure Mapping Engine Implementation*, http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/reasoning/analogy/sme/0.html

Falkenhainer, B, Forbus, K and Gentner, D: 1989, The structure-mapping engine: Algorithm and examples, *Artificial Intelligence* **20**(41): 1-63.

Gentner, D, Holyoak, K and Kokinov, B (eds): 2001, *The Analogical Mind: Perspectives from Cognitive Science*, MIT Press, Cambridge, MA.

Goel, A: 1997, Design, analogy, and creativity, *IEEE Expert* **12**(3): 62-70.

Hart, A: 1986, *Knowledge Acquisition for Expert Systems*, McGraw-Hill, New York.

Kitamura, Y, Kashiwase, M, Fuse, M and Mizoguchi R: 2004, Deployment of an ontological framework of functional design knowledge, *Journal of Advanced Engineering Informatics* **18**(2): 115-127.

Pahl, G and Beitz, W: 2003, *Engineering Design: A Systematic Approach*, Springer, 2nd ed.

Prabhakar, S and Goel, A: 1996, Functional modeling of interactions between devices and their external environments for adaptive design, *Proc. Modeling and Reasoning about Function workshop*, AAAI-96 Conference, Portland, Oregon, pp. 95-106.

Rosenman, MA and Gero, JS: 1998, Purpose and function in design: From the socio-cultural to the techno-physical, *Design Studies* **19**(2): 161-186.

Shaw, LG and Gaines, BR: 2005, Rep IV 1.10. http://repgrid.com, Cobble Hill, BC, Canada

Stone, RB and Chakrabarti, A (eds): 2005, Engineering applications of representations of function, *AIEDAM* **19**(2&3): 359-370.

Stone, RB and Wood, KL: 1999, Development of a functional basis for design, *Proc. ASME Design Theory and Methodology Conference*, DETC99/DTM-8765.

Umeda, Y and Tomiyama, T: 1997, Functional reasoning in design, *IEEE Expert* **12**(2): 42-48.

DESIGN OPERATORS TO SUPPORT ORGANISATIONAL DESIGN

CATHOLIJN M JONKER
University of Nijmegen, The Netherlands

ALEXEI SHARPANSKYKH, JAN TREUR
Vrije Universiteit Amsterdam, The Netherlands

and

PINAR YOLUM
Bogazici University, Turkey

Abstract. Organisational design is an important topic in the literature on organisations. Usually the design principles are addressed informally in this literature. This paper makes a first attempt to formally introduce design operators to formalize the design steps in the process of designing organisations. These operators help an organisation designer create an organisation design from scratch as well as offer the possibility to revise existing designs of organisations. The operators offer both top-down refinements and bottom-up grouping options. Importantly, the operators can be combined into complex operators that can serve as patterns for larger steps in an organisation design process. The usability of the design operators is demonstrated in a running example. This is demonstrated by an implemented prototype example tool.

1. Introduction

Organisations play a key role in the modern society. The welfare of the society as a whole depends upon the effectiveness, efficiency and viability of organisations. Organisational structures and processes are studied in social sciences, where organisational design is a special topic. Organisation design is concerned "with what an organisation is ought to be" (Pfeffer 1978). More specifically, Galbaith (1978) stated that organisation design "is conceived to be a decision process to bring about a coherence between the goals or purposes for which the organisation exists, the patterns of division of labor and interunit coordination and the people who will do the work". Further

J.S. Gero (ed.), Design Computing and Cognition 0'6, 203–222.
© 2006 *Springer. Printed in the Netherlands.*

Galbaith argues that design is an essential process for "creating organisations, which perform better than those, which arise naturally".

In literature, a range of theories and guidelines concerning the design of organisations are present (Galbraith 1978; Duncan 1979; Minzberg 1993; Blau and Schoenherr 1971). However, despite the abundance of organisational design theories no general principles applicable to organisational design in all times and places can be identified (Scott 1998). Moreover, almost all theoretical findings in organisational design are informal and often vague. In order to provide an organisation designer or a manager with operational automated tools for creating, analyzing, and revising organisations, in the first place a formal representation of an organisation model as a design object description should be provided. In addition to this, to address the operations performed on such design object descriptions during a design process, a formal representation of design operators underlying possible design steps is needed. Such design operators describe the possible transitions between design object descriptions. Using the design operators, a design process can be described by, at the various points in time, choosing a next operator to be applied to transform the current design object description into the next one. Examples of very simple design operators are adding or deleting an element of a design object description.

In this paper we introduce a formal organisational model format, to be used to represent design object descriptions. On top of this, a set of design operators is formally defined. The formalisation is based on an extension of predicate logic (Huth and Ryan 2004).

Often in the literature organisational design is recognized as an engineering problem (Child 1973). From this perspective design is considered as a continuous process of a gradual change of an organisational model by applying certain operations (Pfeffer 1978). For example, Minzberg (1993) describes design process as the following sequence of operations: given overall organisational needs, a designer refines the needs into specific tasks, which are further combined into positions. The next step is to build the "superstructure" by performing unit grouping using special guidelines and heuristics (e.g., grouping by knowledge and skill, by work process and function, by time, by place, etc.). Then, the grouping process is repeated recursively, until the organisation hierarchy is complete.

For this paper we aimed at identifying the most commonly and generally used set of operators for designing organisations. For this purpose the literature from social sciences, and design principles used in other disciplines were investigated. Useful principles for organisational design can be found in the area of derivative grammars. Thus, graphical changes in organisational designs may be described by shape (Stiny 1991) and graph grammars (Rozenberg 1997). Whereas changes in textual (or symbolic)

structural and dynamic descriptions of organisational elements may be specified by string (Chomsky 1965) and graph grammars, which allow representation of relationships between descriptions of different elements. In order to relate graphical organisational designs to designs described in a symbolic form, parallel grammars (or grammars defined in multiple algebras) may be used (Stiny 1991). For designing organisation structures with multiple levels of representation (e.g., hierarchical organisations with departments, groups, sections) abstraction grammars (Schmidt and Cagan 1995) and hierarchical graph grammars (Habel and Hoffmann 2004) can be useful. By means of abstraction grammars, design is performed from the top level of the abstraction hierarchy to the bottom (most concrete) level, with each design generation using the prior level design as a pattern. Furthermore, mechanisms for choosing the most appropriate design generated by different transformations defined by grammars have been developed in different areas (e.g. recursive annealing in mechanical design (Schmidt and Cagan 1995)).

Thus, based on the rich literature on design, this paper makes a first attempt to formalize the operators underlying organisation design processes. A set of design operators is formally introduced, which provides the means for creating a design of an organisation from scratch as well as revising existing designs for organisations.

In Section 2 a formal framework for the specification of design object descriptions for organisations is described. Sections 3 and 4 introduce a set of classes of operators to create and modify design object descriptions for organisations. Section 5 illustrates the application of a developed prototype by an example. Finally, Section 6 discusses future work and provides general conclusions.

2. Format for an Organisational Model as a Design Object Description

We consider a generic organisation model, abstracted from the specific instances of agents (actors), which consists only of descriptions of organisational roles and relations between them.

Definition 1 (Organisation)
A specification of an organisation with the name o is described by the relation is_org_described_by(O, □, ▵), where □ is a structural description and ▵ is a description of dynamics.

An organisational structure is characterized by the patterns of relationships or activities in an organisation, and described by sets of roles, groups, interaction and interaction links, relations between them and an environment.

Definition 2 (Organisation Structure)

A structural description Γ of an organisational specification described by the relation is_org_described_by(O, Γ, Δ) is determined by a set of relations, among which[1]:

- a relation has_basic_components(Γ, R, G, IL, ILL, ONT, M, ENV) defined on the subsets R, G, IL, ILL, ONT, M, ENV of the corresponding general sets ROLE (the set of all possible role names), GROUP (the set of all possible group names), INTERACTION_LINK (the set of all possible interaction links names), INTERLEVEL_LINK (the set of all possible interlevel links names), ONTOLOGY (the set of all possible ontology names), ONTO_MAPPING (the set of all possible ontology mappings names), ENVIRONMENT (the set of all possible environment names)2
- a relation for specifying a role r∈R in Γ is_role_in(r, Γ)
- a relation for specifying an interaction link e∈IL in Γ is_interaction_link_in (e, Γ)
- a relation for specifying an interlevel link il∈ILL in Γ is_interlevel_link_in(il, Γ)
- a relation for specifying an environment env∈ENV is_environment_in(env, ENV)
- a relation has_input_ontology(r, o) that assigns an input ontology o∈ONT to a role r∈R (similarly the relations for output, internal, and interaction ontologies are introduced: has_output_ontology(r, o), has_interaction_ontology(r, o), has_internal_ontology(r, o))
- a relation has_input_ontology(env, o) that assigns an input ontology o∈ONT to an environment env∈ENV (similarly the relations for output, internal, and interaction ontologies are introduced: has_output_ontology(env, o), has_interaction_ontology(env, o), has_internal_ontology(env, o))
- a relation is_ontology_for(el, o) that assigns an ontology o∈ONT either to a role el∈ R or an environment el∈ ENV
- a relation has_onto_mapping(il, m) that associates an interlevel link il∈IL with an ontology mapping m∈M (an ontology mapping for an interaction link is defined similarly)
- a relation is_interaction_link_of_type(e, type) that specifies an interaction link e∈IL of one of the types: role_interaction_link, env_input_link, env_output_link

[1] Notice that all the following relations are defined using the names of organization elements; the specifications for these elements will be provided in the following definitions.

[2] The difference between R and ROLE, for example, is that R (subset of ROLE) is the set of all role names that occur in Γ.

- a relation connects_to(e, r, r', □) that specifies a connection by an interaction link e∈IL from a source-role r ∈ R to a destination role r'∈R in □
- a relation connects_to(e, env, r, □) that specifies a connection by an interaction link e∈IL of type env_output_link from an environment env∈ENV to a role r∈R in □ (similarly for connects_to(e, r, env, □))
- a relation subrole_of_in(r', r, □) that specifies a subrole r'∈R of a role r∈R in □
- a relation member_of_in(r ,g ,□) that specifies a member role r∈R of a group g∈G in □
- a relation interlevel_connection(il, r, r', □) that specifies a connection by an interlevel link il∈ILL between roles r, r'∈R of adjacent aggregation levels

Organisational behavior is described by dynamic properties of the organisational structure elements.

Definition 3 (Organisation Dynamics)
A description of dynamics □ of an organisational specification described by the relation is_org_described_by(O, I□, □) is determined by a set of relations, among which:

- a relation has_basic_components(□, DP) that specifies a set of dynamic properties names DP defined in an organisation model
- a relation has_dynamic_property(r, d) that specifies a dynamic property d∈DP for a role r∈R (the relations for dynamic properties of an interlevel link, a group and an environment are defined in a similar manner: has_dynamic_property(e, d), has_dynamic_property(g, d), has_dynamic_property(env, d))
- a relation has_expression(d, expr) that identifies a dynamic property name d∈DP with a dynamic property expression expr∈DPEXPR (e.g., a formula in sorted first-order predicate logic)

A role is a basic structural element of an organisation. It represents a subset of functionalities, performed by an organisation, abstracted from specific agents (or actors) who fulfill them. Each role has an input and an output interface, which facilitate the interaction (communication) with other roles. The interfaces are described in terms of interaction (input and output) ontologies: a vocabulary or a signature specified in order-sorted logic. An ontology contains objects that are typed with sorts, relations, and functions.

Each role can be composed of a number of other roles, until the necessary detailed level of aggregation is achieved. Thus, roles can be specified and analyzed at different aggregation levels, which correspond to different levels of an organisational structure. A role that is composed of (interacting) subroles, is called a composite role.

Definition 4 (Role)
A specification of a role r is determined by:
Objects:

- or, oi, o, o', o''∈ONT, or= o ⊕ o' ⊕ o'', oi= o' ⊕ o'', here ⊕ is a functional symbol that maps names of ontologies to a name of the joint ontology
- Relations:
- has_internal_ontology(r, o), has_input_ontology(r, o'), and has_output_ontology(r, o'')
- has_ontology(r, or) and has_interaction_ontology(r, oi)
- d∈DP, has_dynamic_property(r, d)

The ontologies, which describe interfaces of interacting roles, can be different. Therefore, if necessary, the specification of a role interaction process includes ontology mapping. An ontology mapping m between ontologies o and o' is characterized by a set of relations is_part_of_onto_map(a, a', m), where a is an atom expressed in ontology o and a' is an atom expressed using ontology o'.

Roles of the same aggregation level interact with each other by means of interaction links. The interaction between roles is restricted to communication acts.

Definition 5 (Interaction link)
An interaction link e is determined by:
Relations:

- is_interaction_link_in(e, ⊕)
- has_onto_mapping(e, m) for some m∈M
- has_dynamic_property(e, d) for a number of d∈DP
- Constraints:
- An interaction link e should connect two roles at the same aggregation level: is_interaction_link_in(e, ⊕) ⇒ ∃r, r'∈R connects_to(e, r, r', ⊕) ∧ ¬has_subrole(r, r') ∧ ¬has_subrole(r', r)

An interlevel link connects a composite role with one of its subroles. It represents an information transition between two adjacent aggregation levels. It may describe an ontology mapping for representing mechanisms of information abstraction. For example, consider a situation, in which only a (abstracted) part of information communicated within a certain composite role should be made available as output from this role.

Definition 6 (Interlevel link)
A specification for an interlevel link il is determined by:
Relations:

- is_interlevel_link_in(il, ⊕)

- has_onto_mapping(il, m) for some m∈M
- Constraints:
- An interlevel link il should connect two roles at two adjacent aggregation levels: is_interlevel_link_in(il, □) ⇒ ∃r, r'∈R subrole_of_in(r', r, □) ∧ (interlevel_connection(il, r, r', □) ∨ interlevel_connection(il, r', r, □))

A group is a composite structural element of an organisation that consists of a number of roles. In contrast to roles a group does not have well-defined input and output interfaces. Groups can be used for modeling units of organic organisations, which are characterized by loosely defined or sometimes informal frequently changing structures that operate in a dynamic environment. Furthermore, groups can be used at the intermediate design steps for identifying a collection of roles, which may be further transformed into a composite role.

Definition 7 (Group)
A group g is defined by the relations to other concepts:

- membership relation member_of_in: r∈R member_of_in(r, g, □)
- has_dynamic_property(g, d) for a number of d∈DP

The conceptualized environment represents a special component of an organisation model. According to some sociological theories (e.g., contingency theory), an environment represents a key determinant in organisational design, upon which an organisational model is contingent. Similarly to roles, the environment is represented in this proposal by an element having input and output interfaces, which facilitate in interaction with roles of an organisation. The interfaces are conceptualized by the environment interaction (input and output) ontologies. Interaction links between roles and the environment are indicated in the organisational model as ones that have a specific type, namely env_input_link or env_output_link by means of the predicate is_interaction_link_of_type. Roles interact with the environment by initiating observations and obtaining observation results, and performing actions that can change a state of the environment.

The behavior of each element of an organisational structure is described by a set of dynamic properties. With each name of a dynamic property an expression is associated. Dynamic property expressions represent formulae specified over a certain ontology(ies). In particular, a dynamic property for a role is expressed using a role ontology. A dynamic property for an interaction link is constructed using the output ontology of a role-source of a link and the input ontology of a role-destination. A group dynamic property is expressed using ontologies of roles- members of a group. An example of the dynamic property expression will be given in Section 3.1.

The application of the basic components of an organisational model is illustrated by means of a running example. Consider the process of

organizing a conference. A partial model for the considered conference organisation is shown in Figure 1.

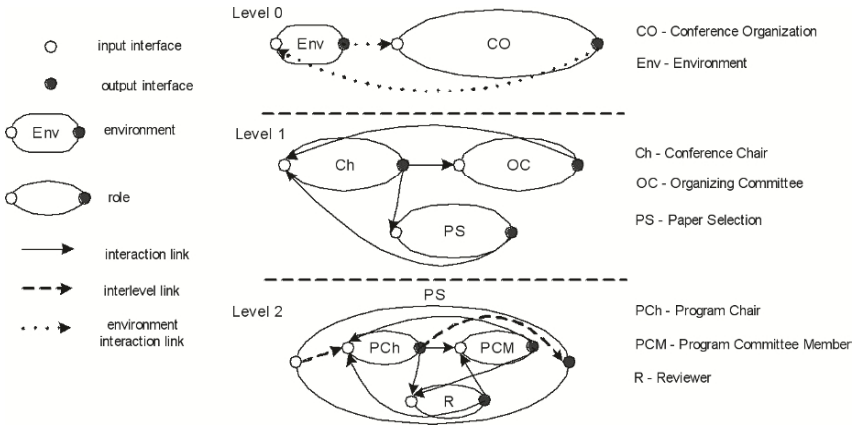


Figure 1. Model of the conference organizing committee.

At the most abstract level 0 the organisation is specified by one role CO (Conference Organisation) that interacts with the environment Env. Role CO can act in the environment, for example by posting a call for papers in different media. Note, that the organisational model is depicted in a modular way; i.e., components of every aggregation level can be visualized and analyzed both separately and in relation to each other. Consequently, scalability of graphical representation of an organisational model is achieved. At the first aggregation level the internal structure of the composite role CO is revealed. It consists of subrole Ch (Conference Chair), which interacts with two other subroles: OC (Organizing Committee) and PS (Paper Selection role). At the second aggregation level the internal structure of role PS is represented. It consists of subrole PCh (Program Chair), subrole PCM (Program Committee Member), and subrole R (Reviewer), which interact with each other. The input interface of role PS is connected to the input interface of its subrole PCh by means of an interlevel link. In our example the interlevel link describes the mapping between the input ontology of role PS and the input ontology of its subrole PCh. It means that information, transmitted to the role PS at the first aggregation level, will immediately appear at the input interface of subrole PCh, expressed in terms of its input ontology at the second aggregation level.

3. Representing Design Operators for Organisational Design

In this section a formal format to represent design operators and based on this format representations are introduces for a number of primitive design operators for designing organisations. Each primitive operator represents a

specialized one-step operator to transform a design object description (organisational model) into a next one. The parts of the organisation O that are being modified in terms of structure and dynamics (i.e., sets of dynamic properties) are specified using the in-focus relations: structure_in_focus(O, Rf, Gf, ILf, ILLf, ONTf, Mf, ENVf) and dynamics_in_focus(O, DPf), with Rf⊆R, Gf⊆G, ILf⊆IL, ILLf⊆ILL, ONTf⊆ONT, Mf⊆M, ENVf⊆ENV, DPf⊆DP. The remaining parts of the organisation stay the same.

The following operations all refer to an organisation O∈ORGANISATION described by relations is_org_described_by(O, Ω, Δ), has_basic_components(Ω, R, G, IL, ILL, ONT, M, ENV). This organisation is modified by an operator, leading to a second organisation O'∈ORGANISATION described by relations is_org_described_by(O', Ω', Δ'), has_basic_components(Ω', R', G', IL', ILL', ONT', M', ENV').

Our choice of primitive operators is motivated by different design guidelines and theories from social sciences (Galbraith 1978; Blau and Schoenherr 1971; Lorsch and Lawrence 1970), other disciplines, and our own research on formal modeling of organisations (Broek et al. 2005). However, the application of the proposed set of operators is not restricted only to these theories. Thus, a designer has freedom to choose any sequence of operators for creating models of organisations. The operators are divided into three classes, which are consecutively described in the following subsections. Thus, in Section 3.1 the operators for creating and modifying roles are specified; in Section 3.2 the operators for introducing and modifying different types of links are described; and in Section 3.3 the operators for composing and modifying groups are introduced.

3.1. OPERATORS FOR ROLES

The classes of primitive operators for creating and modifying roles in a design object description for an organisation are shown in Table 1.

TABLE 1. Operator classes for creating and modifying roles.

CLASS	DESCRIPTION
Role Introduction	Introduces a new role
Role Retraction	Deletes all links, connected to a role with their dynamic properties and mappings; deletes a role and all dynamic properties, associated with this role
Role Dynamic Property Addition	Adds a new dynamic property to a role
Role Dynamic Property Revocation	Deletes an existing role dynamic property

A *role introduction operator* adds a new role to the organisation. Usually, in organisational design after organisational tasks have been identified, these tasks should be further combined into positions (roles), based on the labor division principles (Kilbridge and Wester 1966).

Role introduction operator

Let op(O, O', δ) be an operator that changes O into O' with a focus on δ. Then op is a role introduction operator iff it satisfies:

1. δ ∉ R, δ ∈ R' such that is_role_in(δ, □')
2. structure_in_focus(O, ∅, ∅, ∅, ∅, ∅, ∅, ∅)
3. structure_in_focus(O', {δ}, ∅, ∅, ∅, ONT', ∅, ∅), where ONT'= is_ontology_for(o, δ) and o ∈ ONT'

A *role retraction operator* removes all links, connected to a role with their dynamic properties and mappings; it also deletes dynamic properties, associated with the role and the role itself. In the example of the conference organisation, when the Reviewer Recruiter has found enough reviewers, then the role can safely be removed from the organisation.

A formal representation for the role retraction operator has been left out due to the limited space and can be found in Jonker et al. (2005).

A *role dynamic property addition operator* creates a new property for the existing role in the organisation. For example, a role property that may be added to role Reviewer (R) expresses that a reviewer should send her review to the Program Chair before a certain deadline. This property can be formalized using the Temporal Trace Language (TTL) (Jonker and Treur 2003a), which is a variant of an order-sorted predicate logic with facilities for reasoning about the dynamics properties of a system. Thus, the dynamic part of the organisational model is changed by adding the following dynamic property for role R:

∀t state(∅, t) |= deadline_for_conference(d) ⇒ ∃t' < d state(∅, t', output(Reviewer)) |= communicated(send_from_to(Reviewer, Program_Chair, review_report))

A *role dynamic property revocation operator* deletes a property from the dynamic description of a role.

3.2. OPERATORS FOR LINKS

In this subsection, we propose a set of classes of primitive operators for creating and modifying links in a design object description for an organisation, Table 2.

TABLE 2. Operator classes for creating and modifying links.

CLASS	DESCRIPTION
Interaction Link Addition	Adds a new interaction link between any two roles
Interaction Link Deletion	Deletes an interaction link and all dynamic properties, associated with this link
Interlevel Link Introduction	Introduces a new interlevel link
Interlevel Link Retraction	Retracts an existing interlevel link
Interaction Dynamic Property Addition	Adds a new dynamic property to an interaction link
Interaction Dynamic Property Revocation	Deletes an existing dynamic property, associated with an interaction link

An *interaction link addition operator* allows the creation of an interaction link (information channel) between two existing roles in the organisation. In the organisational design after organisational subtasks are assigned to roles, the problem of coordination of interdependencies among subtasks should be solved.

In the conference management example, the Program Chair (playing in this case a managerial role) may request two reviewers to discuss their reviews. This requirement can be handled by the addition of an interaction link between the appropriate reviewer roles in the design object description for an organisation.

Interaction link addition operator
Let op(O, O', δ) be an operator that changes O into O' with a focus on δ. Then op is an interaction link addition operator iff it satisfies:

1. ∃δ∉IL,δ∈IL' such that is_interaction_link_in(δ, □')
2. structure_in_focus(O,∅,∅,∅,∅,∅,∅,∅)
3. structure_in_focus(O', ∅, ∅, {δ}, ∅, ∅, Mf', ∅)
   Mf'= {m∈M'| has_onto_mapping(δ, m)}

An *interaction link deletion operator* is used to delete an existing interaction link between two roles as well as to revoke all dynamic properties, associated with this link. For example, the Program Chair has taken care of the acceptance proceedings for the conference. He does not need to be in contact with the reviewers any more. This case can be handled by the deletion of the interaction between two roles in the design object description for an organisation.

An *interaction property addition operator* creates a new property for an existing interaction link. An *interaction property revocation operator* deletes a property from the dynamic description of an interaction link.

An interlevel link creates a relation between a composite role and its subroles. It allows information that is generated outside the role, to be passed into the role through its input interface or it allows information, generated within a role to be transmitted outside through the role output interface. Normally, in hierarchical (mechanical) organisations decisions made at a managerial level are transferred to an operational level, e.g, to a certain department. Within the department this information is obtained by a certain role(s). For identifying, which roles obtain this information interlevel links are used. In the conference management example, the Conference Chair may have the possibility to send inquiries to Program Committee Members. This can be achieved by introduction of an interlevel link between composite role Paper Selection (with which role Conference Chair has a direct connection by an interaction link) and its subrole Program Committee Member. An *interlevel link introduction operator* allows addition of such a link into a role.

Interlevel link introduction operator
Let op(O, O', δ) be an operator that changes O into O' with a focus on δ. Then op is an interlevel link introduction operator iff it satisfies:

1. ∄δ∈IL,δ∈IL' such that is_interaction_link_in(δ, □')
2. structure_in_focus(O,∅,∅,∅,∅,∅,∅,∅)
3. structure_in_focus(O',∅,∅,{δ},∅,∅, Mf',∅)
   Mf'= {m∈M'| has_onto_mapping(δ, m)}

An *interlevel link retraction operator* is used for breaking off interaction between some composite role and one of its subroles. This operation removes an interlevel link from the design object description for an organisation. If the Conference Chair does not need to communicate with Program Committee Members any more, the interlevel link between these two roles can be retracted.

3.3. OPERATORS FOR GROUPS

The classes of primitive operators for creating and modifying groups in a design object description for an organisation are shown in Table 3.

Often an organisation designer can easily list a number of roles needed in an organisation. However, it is not always clear, which roles are related to each other; which roles would most often interact with each other, and so on.

TABLE 3. Operator classes for creating and modifying groups.

CLASS	DESCRIPTION
Grouping	Combines roles into groups
Degrouping	Moves roles outside of a group and deletes the group
Group-to-Role Transformation	Transforms groups into roles
Role-to-Group Transformation	Transforms roles into groups

In the literature on organisational design (Minzberg 1993) different principles of grouping are described. For example, role grouping can be performed based on (1) similarities in role functional descriptions; (2) role participation in the same technological process; (3) identity or similarity of role technical specialties; (4) role orientation on the same market or customer groups. Often roles belonging to the same group interact with each other intensively. However, in the proposed organisational model in contrast to roles, groups do not have interfaces. It means that every role within a group is allowed to interact with roles outside the group by means of direct interaction links. A group can be transformed into a role, a more coherent, integrated and formal organisational unit with proper interfaces (e.g., a department of an organisation). For example, in the conference organisation the Program Chair and the Program Committee Members can be joined in one Program Committee group that will be responsible for making final

decisions concerning paper acceptance. This can be accomplished by applying the grouping operator.

Grouping operator

Let op(O, Rg, O', Gn) be an operator that changes O into O' wrt. Gn∉G', Rg⊆R. Then op is a grouping operator that creates a new group Gn from the subset of roles Rg iff it satisfies:

Structural aspect:

1. ∀a∈Rg: member_of_in(a, Gn, □').
2. structure_in_focus(O, ∅, ∅, ∅, ∅, ∅, ∅, ∅)
3. structure_in_focus(O', ∅, {Gn}, ∅, ∅, ∅, ∅, ∅)

Dynamic aspect:

1. dynamics_in_focus(O, ∅)
2. dynamics_in_focus(O', DPf')
   DPf'={dp∈DP'| has_dynamic_property(Gn, dp) }.
3. Er={e∈IL| ∃r1∈Rg ∃r2∈Rg connects_to(e, r1, r2, □)}
   DPr={dp∈DP|∃r∈Rg has_dynamic_property(r, dp) ∨∃e∈Er has_dynamic_property(e, dp)}
   DPg={dp∈DP'| has_dynamic_property(Gn, dp)}
4. DPg⊆DCL(DPr), where DCL(DPr) is the deductive closure of DPr

A natural dual to the role grouping is role degrouping. This operator takes a group of roles and moves the roles to outside of the group. Role Degrouping transforms a group into a set of roles.

For a group to act as a role, it should have well-defined (formalized) input and output interfaces. A Group-To-Role operator takes a group and adds these interfaces. In an organic organisation with loosely defined frequently changing structure this would correspond to the formalisation of one of the organisational units, i.e., providing a formal (permanent) structural description with the subsequent specifying formal functional procedures. For example, in the conference organisation setting Program Committee group from the Paper Selection role can be further transformed into Program Committee role, a formal organisational unit with certain characteristics and functions (e.g., final decision making for the paper acceptance). In this case reviewers should follow a formal procedure for interactions with Program Committee role and cannot directly address any arbitrary Program Committee member. Such transformation can be achieved by means of Group-to-Role operator.

Group-to-Role operator

Let op(O, g, O', r) be an operator that transforms group g∈G in O into role r∉R' in O'. Then op is a group-to-role operator iff it satisfies:

Structural aspect:

1. r∉R, g∉G'.
2. ∀a∈R: member_of_in(a,g, □)⇒subrole_of_in(a, r, □').
3. structure_in_focus(O, ∅, {g}, ∅, ∅, ∅, ∅, ∅)
4. structure_in_focus(O', {r}, ∅, ∅, ∅, ONTf', ∅, ∅)
   ONTf'={o∈ONT'| has_internal_ontology(r, o) ∨ has_input_ontology(r, o) ∨ has_output_ontology(r, o)}

Dynamic aspect:

1. dynamics_in_focus(O, DPf)
   DPf={dp∈DP| has_dynamic_property(g, dp)}.
2. dynamics_in_focus(O', DPf')
   DPf'={dp∈DP'| has_dynamic_property(r, dp)}.
3. DP(g)⇉DP(r)

A role may consist of several other roles that are not exposed to the rest of the world. When a role is converted to a group, it exposes the input and output interfaces of the roles inside it. Transforming a role into a group results in the subroles now residing on the level of the prior composite role. For example, during the reorganisation some formal organisation units (e.g., groups, sections, and departments) have been eliminated, whereas the roles that constituted these units and relations between them were kept, thus, creating a basis for new organisational formations.

4. Composing Operators

The described above primitive operators reflect major principles of organisational design. In practice next to the primitive operators more complex operators are used. Complex operators are represented as a combination of primitive operators; some of them are given in Table 4.

Sometimes an effect produced by application of some composite operator to a design object description for an organisation can be achieved by different combinations of primitive operators.

Consider the Role Refinement operator as an example. This operator divides a role into several roles such that the role properties of the first role are distributed over the newer roles. In organisational design role refinement corresponds to the fine-tuned specialization and division of labor for increasing efficiency. It is usually recommended to divide the work so that the portions be differentiated rather than similar, and that each role is responsible for a small portion of the overall task. According to Adam Smith, division of labor is limited by the extent of the market; other general principles of labor division can be found in (Kilbridge and Wester 1966).

Let us illustrate the application of Role Refinement operator in the context of the conference organizing example. In Figure 2 the design object description for an organisation is represented at the first aggregation level. The symbol * denotes that an operator can be applied zero, one or multiple times.

Consider the situation when the decision is made to divide the tasks of Organizing Committee (OC) between the Local Organizing Committee (LOC), which is hence responsible for negotiations with publishers for printing proceedings and arranging the conference venue, and the General Organizing Committee (GOC), which is designated for solving financial and other questions. Thus, role OC is refined into two newer roles LOC and GOC. These roles are able to interact with each other and with role Chair.

TABLE 4. Sample complex operators for creating and manipulating organisations.

NAME	PATTERN FOR	DESCRIPTION
Interaction Level Ascent	Interaction link deletion*. Role interaction dynamic property addition*. Interlevel link addition*. Interaction link addition*.	Represents interaction between roles at a higher aggregation level
Role refinement	Role Retraction. Interlevel link deletion*. Interaction link deletion*. Interaction dynamic property addition*. Interlevel link addition*. Interaction link introduction*. Role dynamic property addition*. Role introduction*	Divides a role into several roles such that the role properties of the first role are distributed over the newer roles
Role join	Role Retraction*. Interlevel link deletion*. Interaction link deletion*. Interaction dynamic property addition*. Interlevel link addition*. Interaction link introduction*. Role dynamic property addition*. Role introduction	Joins several roles into a single role
Adding aggregation levels	Interaction Level Ascent. G-t-R. Role grouping. Role refinement*	Aggregates existing roles of the organisation in more complex roles

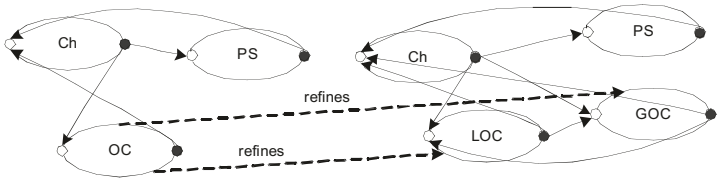


Figure 2. Example of Role refinement operator application, in which the Organizing Committee role (OC) is refined into the Local Organizing Committee (LOC) and General Organizing Committee roles (GOC).

Alternatively, every composite operator can be considered as an aggregated one-step operator. Such descriptions define formal conditions for a design object description for an organisation before and after the application of a complex operator; therefore, they can serve for the purposes of checking integrity and consistency of a design object description.

A natural dual to the role refinement is role joining. This operator takes several roles and joins them into a single role. Consider again the organisation arranging a conference. If over time the differences between the tasks of the Program Committee Member and Reviewer roles disappear, then the roles Program Committee Member and Reviewer can be joined in one role.

Let us consider one more often used complex operator *Adding Aggregation Levels*. When certain roles have been joined in one group, this operator allows representing this group as an integral structural unit of an organisation at the more abstract aggregation level. This operator has a counterpart in organisational design studies called *departmentalization*. Based on the departmentalization principles (cf. Galbraith 1978) an organisation is partitioned into structural units (called departments) with certain areas of responsibilities, a functional orientation, and a local authority power.

In the conference organisation Adding Aggregation Levels operator can be applied for representing the Program Committee as an integral role that consists of the Program Chair and the Program Committee Member roles within Paper Selection role. Such choice, for example, can be motivated by introducing a general formal procedure for paper acceptance. Hence, the Program Committee role is empowered (has a corresponding dynamic property) to make final decisions concerning paper selection. Adding Aggregation Levels operator for this example can be considered as three-step process (see Figure 3 for the representation of the organisation model (role Paper Selection) at the second aggregation level).

First, roles Program Chair (PCh) and Program Committee Member (PCM) are joined into one group by application of Grouping operator. After that, at step 2 by means of the Group-to-Role operator the created group is transformed into role Program Committee by adding interaction interfaces. Finally, as the last step using Interaction Level Ascent operator interaction links between roles PC and Reviewer (R) are created, as well as interlevel links within role PC.

5. A Prototype Tool to Support the Design of Organisations

The formal representations of the organisation entities and the design operators described in this paper provide a solid basis for the development of a software environment supporting interactive organisation design processes.

The proposed formalism accurately distinguishes different types of organisation entities with their objects, relations and constraints, which can be naturally represented as classes with members and methods in object-oriented programming (OOP) languages. Furthermore, the identified relationships among organisation entities may be fully captured by the fundamental OOP mechanisms (e.g., inheritance, interfaces and inner classes). The design operators can be programmed as transformation functions with explicitly defined arguments, conditions and effects of their application. Moreover, most of the introduced formal concepts are based on the notions from organisation theories, which will facilitate use of a tool by organisation modelers.

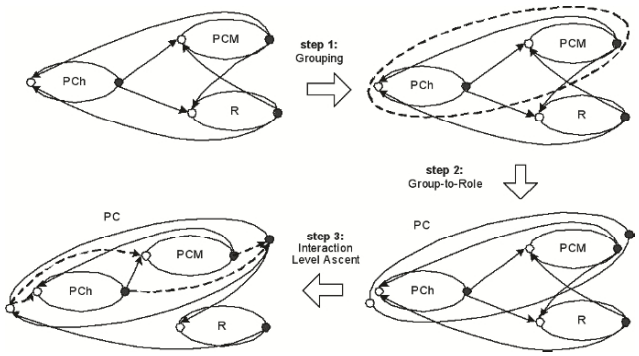


Figure 3. Example of Adding Aggregation Levels operator application, in which the roles Program Chair (PCh) and Program Committee Member (PCM) are grouped together and transformed into the Paper Selection (PC) role.

For the purpose of illustration and evaluation a prototype tool was implemented. This tool supports organisational design and allows investigating its dynamics. The application of the design prototype is demonstrated on the example of role refinement as described in the previous Section. The dynamics of the design process is described in Table 5, which is graphically illustrated by a partial trace taken from the tool in Figure 4.

In the design process, first, a designer chooses a part of the design object description, on which she intends to put her attention (in the considered example it is the role Organizing Committee). Next, the software proposes to the designer a number of operators, which are potentially applicable to the chosen part of the design object description. The designer chooses one of them, for the example, the role refinement operator. Refinement is a composite operator that consists of an ordered sequence of primitive operators. Usually, most of the primitive operators constituting composite ones are imperative (e.g., Role Introduction for Refinement); yet application of some of them may be postponed to the future (e.g., Role dynamic property addition for Refinement) or skipped (e.g., Interlevel link deletion for Refinement). Further, the tool demands specifying roles, into which role OC has to be refined. The designer specifies role names (for this example, Local Organizing Committee (LOC) and General Organizing Committee (GOC)) and their ontologies.

After that the software tool requests the designer to specify dynamic properties for the created roles. The designer may postpone this operation to a future time point. Thereafter, the tool proposes to add interaction links between roles LOC, GOC and role Chair (Ch), with which the original role OC was connected. After that dynamic properties for the introduced interaction links may be added. As the last step role OC and interaction links

connecting it with role Ch, as well as OC role and interaction links dynamic properties are automatically removed from the design object description.

TABLE 5. Dynamics of the design process for the role refinement.

ACTIONS OF THE DESIGNER	STATES OF THE TOOL
Chooses to address the role Organizing Committee (OC)	Proposes potentially applicable operators for role OC
Chooses the role refinement operator	According to the specification of the role refinement operator, initiates execution of role introduction operator and requests the designer to specify role names
Specifies GOC (General Organizing Committee) and LOC (Local Organizing Committee) names of the roles, into which role OC is refined	Requests to specify the elements of the ontologies for the newly created roles
Specifies the elements of the ontologies for roles LOC and GOC	Initiates execution of the role dynamic property addition operator. Requests to specify dynamic properties for LOC and GOC roles
(optional) Specifies dynamic properties for the roles	Initiates execution of the interaction link introduction operator. Requests to specify interaction links between roles Chair (Ch), LOC and GOC
Specifies, which interaction links are needed between the roles	Initiates execution of the interaction dynamic property addition operator. Requests to specify dynamic properties for the introduced interaction links
(optional) Specifies dynamic properties for the interaction links	Initiates execution of the interaction link deletion operator, which removes all interaction links connected with role OC. Then, initiates execution of the role retraction operator, which removes role OC from the design object description

6. Discussion

This paper introduces a representation format and a variety of operators for the design of organisations specified in this representation format. The described operators have several important characteristics. First, they can be combined into composite operators that can serve as patterns for larger design steps in certain design cases. Second, the identified set of operators is independent of any organisation theory or sociological methodology: they can be used for formalizing design principles from different theories. Third, a designer has freedom to choose any sequence of operators for creating designs of organisations of most types (e.g., functional and organic). The

operators offer both top-down refinements, as well as bottom-up grouping options. Finally, as has been shown the developed tool provides interactive support in designing organisations. In the future a graphical interface for representing design objects in the developed tool will be developed.

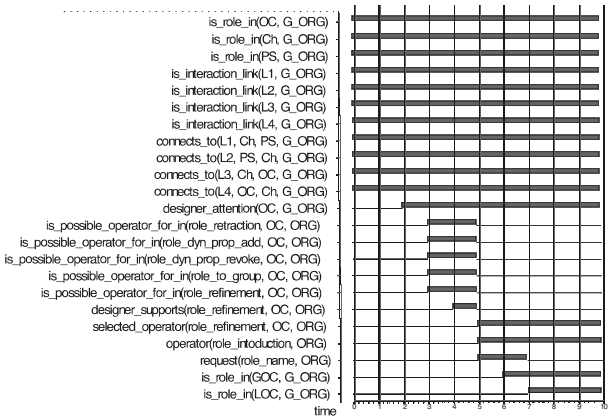


Figure 4. Screen print of a trace illustrating dynamics of the design process for the role refinement.

In the area of component-based software engineering a number of design patterns for building software components (e.g., refinement, chaining, disjoint composition) have been introduced (He et al. 2005). These patterns specify general-purpose manipulations with programming constructs (e.g., interface and private methods of components); while in organisational design literature organisation transformations are described using domain-specific concepts. The formal representation format proposed in this paper bridges this gap and facilitates the abstraction of organisation domain into general-purpose programming design patterns.

Formal specification of design processes enables verification of structural and dynamic consistency of a design object description for an organisation. The verification of structural consistency is based on the consistency definitions for operators (Jonker et al. 2005). For verifying dynamic consistency model checking techniques (McMillan 1993) may be used, which will be further investigated in the future. Furthermore, verification mechanisms based on certain requirements on organisational functioning and performance (e.g., using organisation performance indicators) represent a subject of our future research.

In conclusion, this paper introduced a representation format and a set of formally represented design operators dedicated to the design of organisations of most types. Although the choice of operators is motivated by different theories and guidelines from the area of organisational design,

the application of the proposed operators is not restricted to any theories from social studies. The formalisation of the operators provides a solid basis for the development of a software tool supporting interactive organisation design processes. A prototype implementation for such a tool is demonstrated by an example in this paper.

Acknowledgements

The authors wish to thank the anonymous reviewers for their useful comments. Their comments also found their way into the technical report upon which this paper is based.

References

Blau, PM and Schoenherr, RA: 1971, *The Structure of Organisations*, Basic Books Inc., New York London.

Broek, E, Jonker, C, Sharpanskykh, A, Treur, J, and Yolum, P: 2005, Formal modeling and analysis of organisations, *in* O Boissier, V Dignum, E Matson, J Sichman (eds), *Proceedings of the Workshop on Organisations in Multi-Agent Systems*, pp. 17-33.

Child, J: 1973, *Organisation: A Choice for Man*, *in* J Child (ed), *Man and Organisation*, Halsted Press, London, pp. 234-570.

Chomsky, N: 1965, *Aspects of the Theory of Syntax*, The MIT Press.

Duncan, RB: 1979, What is the right organisation Structure? *Organisational Dynamics*, Winter, pp. 59-79.

Galbraith, JR: 1978: *Organisation Design*, Addison-Wesley Publishing Company, London Amsterdam Sydney.

Habel, A and Hoffmann, B: 2004, Parallel independence in hierarchical graph transformation, *International Conference on Graph Transformation*, LNCS, Springer-Verlag, Heidelberg **3256**: 178-193.

He, J, Li, X, and Liu, Z: 2005, Component-based software engineering, *in* DV Hung, M Wirsing (eds), *Theoretical Aspects of Computing*, LNCS, Springer **3722**: 70-95.

Huth, M and Ryan, MD: 2004, *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press.

Jonker, CM, Treur J: 2003, A temporal-interactivist perspective on the dynamics of mental states, *Cognitive Systems Research Journal* **4**(3): 137-155.

Jonker, CM, Sharpanskykh, A, Treur, J, and Yolum, P: 2005, *Operators for Formal Modeling of Organisations*, Technical report 06-01AI, Vrije Universiteit, Amsterdam.

Kilbridge, M and Wester, L: 1966, An economic model for the division of labor, *Management Science* **12**(6): 255-269.

Lorsch, JW and Lawrence, PR: 1970, *Organisation Design*, D Richard (ed), Irwin Inc, USA.

McMillan, K: 1993, *Symbolic Model Checking*, Kluwer Academic Publishers.

Mintzberg, H: 1993, *Structure in Fives: Designing Effective Organisations*, Prentice-Hall, NJ.

Pfeffer, J: 1978, *Organisational Design*, AHM Publishing Corp., Illinois, USA.

Rozenberg, G (ed): 1997, *Handbook of Graph Grammars and Computing by Graph Transformation*, **1**: Foundations, World Scientific.

Schmidt, LC and Cagan, J: 1995, Recursive annealing: A computational model for machine design, *Research in Engineering Design* **7**: 102-125.

Scott, WR: 1998, *Organisations: Rational, Natural and Open Systems*, Prentice Hall, USA.

Stiny, G: 1991, The algebras of design, *Research in Engineering Design* **2**: 171-181.

Wijngaards, N: 1999, *Re-design of Compositional Systems*, PhD Thesis, SIKS dissertation Series, 99-6, Vrije Universiteit Amsterdam.

BAYESIAN NETWORKS FOR DESIGN

A stochastic design search method

PETER MATTHEWS
Durham University, UK

Abstract. A method for flexibly searching the conceptual design space using a stochastic approach is presented. From a database of previous design exemplars, a novel and inexpensive algorithm is used to induce a Bayesian Belief Network (BBN) that represents the causal relationships between a design domain's variables. This BBN is then used as part of an interactive tool for stochastically searching the conceptual design space using two search heuristics. This method is illustrated using a number of design scenarios based on a conceptual car design domain. The paper concludes with future research avenues to further the functionality of the BBN-based design search tool.

1. Introduction

The conceptual design stage occurs during the earliest parts of the design process. This is where a design specification is transformed into an abstract solution, representing the core concepts of the final design. The fluid nature of the conceptual design stage provides a challenge when developing deterministic models of a design at this phase. Specifically, it is difficult to explicitly define metrics for concept quality and this is left to the subjective expertise of the design team. The nature of conceptual design means that it is possible for a 'good' concept to be poorly detailed and thus result in a poor final product and *vice versa*. However, in general good concepts are more readily transformed into good final products while poor concepts require greater effort to attain a similar final high quality level.

A potential approach to this challenge is to adopt a stochastic perspective of the conceptual design phase. This allows for a more flexible representation of the design domain where multiple outcomes are possible. By using Bayesian Belief Networks (BBNs) to model a design domain, it is possible to work with partially defined design concepts. As more of the design is specified, the more accurate the model becomes at predicting how the remainder of the design is likely to be. An interesting and powerful

J.S. Gero (ed.), Design Computing and Cognition '06, 223–241.
© 2006 *Springer. Printed in the Netherlands.*

aspect of the BBN is that it does not distinguish between the design parameters that are directly controlled by the designer and design characteristics which are determined as a result of the designer's decisions on the design parameters. This allows a designer to specify the characteristics at the outset and to then be guided towards design parameters that are likely to secure these characteristics.

This research has developed a method for inducing a BBN from a database of prior design exemplars using a novel information metric (Section 4). Once the BBN has been instantiated, a set of search heuristics are proposed to help guide a designer using the BBN to complete a partial conceptual design (Section 5). This method is illustrated using a set of design scenarios (Section 6). The paper concludes with a discussion of this method and some future development avenues for this stochastic approach.

2. Background

The first task in the design process can be argued as determining the specification of the final constructed artefact or product. The specification will be a combination of 'demands' that the design must fulfil and weighted wishes, which represent desirable but not essential aspects of the design. This specification can be expressed as a simple list of features (Pugh 1990) or encoded as an 'acceptability function' (Wallace et al. 1996). The specification guides the designer towards generating concepts that fulfil the demands. Alternative designs are discriminated between how well they either fulfil the wishes or evaluate against the acceptability function. Provided the specification does not impose overly restrictive demands, the designer is still left with a large conceptual design space to explore.

Conceptual design is by definition fluid. It is left to the detail and embodiment stages to crystalise the design into an artefact that can be manufactured (Pahl and Beitz 1996). A good concept will be easily transformed into a good final design. Conversely, a poor concept will require extensive effort to transform into a good final design. This definition of good/bad concept can only be measured after the final product has been produced, and is of little use during the conceptual stage of the design process. Also, the notion of a 'good' final design is domain and context sensitive. A designer will have a notion of what aspects of the final design are desirable, and a good designer will create concepts that are more likely to have these outcomes.

As a means for resolving the lack of explicit overall quality measure, an alternative, stochastic, view is adopted. This stochastic approach is fundamentally that a good concept has a high probability of resulting in a good final design, whereas a poor concept has a low probability of being transformed into a good design. This leads to a stochastic view of the design

process: the probability of a good design at the end of the process depends on the quality of the initial design concept.

The fluidity of the conceptual design phase means it is difficult to provide concrete evaluation tools. Methods exist for creating 'robust' designs and, through objective evaluation techniques, guide the designer towards concepts that will be able to tolerate changes in the original specification (Taguchi et al. 1989; Ziv-Av and Reich 2005). In effect, these methods aim to provide the most generic design solution that is acceptable. These methods require a predefined evaluation function for the design that encodes the original design specification. An alternative stochastically driven approach is to bias towards design refinement that do not have 'spiky' probability distribution functions (PDFs). Such PDFs lack robustness as any deviation from the peak will result in a significant reduction in the likelihood of design success.

The approach taken in this paper is to provide guidance on the order that design variables should be determined. This designer guidance concept is similar to the Signposting methodology (Clarkson and Hamilton 2000), however it uses the shape of the dynamically computed PDFs rather than predefined domain rules to determine the order that the design variables should be determined.

An important aspect of this method is the inducing of domain models from previous design exemplars. The methods for creation of domain models can be represented on a spectrum ranging from expert based through to fully algorithmic. The expert based end of the spectrum provides high quality transparent models, however these require considerable time investment from domain experts which can be prohibitive. At the other extreme, pure machine learning methods tend to provide complex and opaque models, which while accurate, do not necessarily provide a designer with significant insight into the domain.

A motivating factor for this research is the cognitive aspects that affect human designers. These include the range of model complexities that can be intuitively handled; the nature of understanding a design domain; the latent differences between novice and expert designers; and what constitutes an intuitive interface to a stochastically based design domain model.

3. Bayesian Design

Bayesian design is the use of Bayesian Belief Networks to support the design process. Bayesian Belief Networks (BBNs) provide a causal model for a set of observations or variables (Jensen 2001). These models are represented graphically, where the observations are the graph nodes and the causal links are the directed edges that connect the nodes. As the networks tend to be relatively sparse, namely that nodes are typically only attached to

a small subset of other nodes, this significantly simplifies the computational effort required to make inferences given a set of observations. As observations are made, these provide information for the model. The model uses these observations to make informed estimates on the values of the non-observed variables. For a non-observed variable, it is possible to compute its informed (conditional) probability distribution function. Effectively, the available information biases the unobserved variable's PDF.

In the design context, the observed variables are the design parameters and characteristics. The distinction between these is primarily that design parameters are directly determined by the designer while design characteristics are a result of the design parameters. For the purposes of this work, no distinction is made between these two, as it is impossible in general to infer the causal order between the design variables. For example, when designing a bridge one of the design parameters is the width of the bridge. The wider the bridge, the greater the potential flow across the bridge which is a design characteristic of the bridge. However, a greater potential flow across the bridge will require a stronger bridge, which can be achieved through a number of alternatives, e.g. material choice, structural design, etc., all of which are design parameters again.

Bayesian design is a stochastic view of design, and is particularly appropriate for routine early design tasks. Due to the fluid nature of the early design phases, this is an appropriate approach. Under the stochastic view, each design variable has a PDF. This PDF is a mapping from the values the design variable can take (design space) to the probability of that variable taking that value. The probability of a variable taking on a particular value represents is a measure of how frequently that variable takes that value in final (e.g. detail phase) designs. This can be interpreted as a measure of the design knowledge or experience that exists for achieving the given design variable value. Thus, where low probabilities are encountered, this provides a warning that a potential challenge lies ahead in achieving that position in the design space.

As these PDFs are computed within a BBN, these will be biased where relevant information is available. Relevant information in this context are observations taken from neighbouring nodes within the network. The updated conditional PDFs (CPDFs) now take into account the knowledge that exists about a subset of designs from the domain, as defined by the relevant information that has been added. So where previously setting a design variable to particular value might have appeared difficult to achieve by nature of the low probability of this outcome, it is possible that given the additional information this is becomes a much more likely outcome.

This leads into exploiting design BBN as a design support tool. A designer will start with a specification that defines a subset of the design variables. These defined variables can be considered as observations and

thus be entered into the BBN. The BBN can now provide CPDFs for the unobserved variables. These unobserved variables were not part of the specification, and hence it may be assumed that the designer is free to set these arbitrarily. The designer wishes to produce a design concept that will have the greatest chance of producing a good concept, as these are least likely to require extensive effort during the detailing phases to produce a good final design. Hence, the designer should be attracted to set design variables to their most likely states, as these represent the states where the most knowledge and/or experience exists.

Where a number of different variables require determining, a simple ordering heuristic can be applied. Design variables with narrow 'spiky' distributions should be determined first, proceeding through until the variables with the 'flattest' PDFs being last. This ensures that design variables with narrow likely ranges are set suitably as early as possible. If this is not done, it is possible that through the setting of another design variable, the 'narrow' design CPDF disappears altogether, thus representing a highly unlikely design. In effect, this is the stochastic equivalent of over constraining a design. Similarly, the 'flat' PDFs are likely to become spikier as more of the design is defined. By monitoring how each individual PDF changes with each additional design variable setting, it is possible to dynamically guide a designer through the order in which the design variables should be set. It is worth noting, however, that these are no more than guiding heuristics. Designers are at liberty to navigate through the design domain based on their personal experience or instincts.

4. Inducing Bayesian Networks

To use Bayesian Belief Networks as a design support tool, it is essential to acquire a good BBN in the first instance. The first step to achieve this is the creation of a suitable representation or encoding of the design domain. This provides a definition of the conceptual design space of the domain under consideration. A simple, but suitable, representation format is a design vector. The design parameters and characteristics form the variable components of the vector. As discussed in the previous section, these are to be the nodes of the BBN.

The next step is identifying the causal links between these design variable nodes. One method for achieving this is to use an expert (or panel of experts) to manually identify the links. While this is expected to produce accurate models, it is a time consuming exercise. As the domain becomes more complex in terms of number of design variables, the complexity of the model creation increases quadratically with the number of design variables. Further, once the nodes have been linked, the PDFs and CPDFs that are associated with the nodes and arcs respectively must be defined. This

requires considerably greater consideration than identifying the causal links. As a result, the expert crafted BBN is not appealing.

An alternative method for identifying the causal links in the BBN is to apply data mining techniques to a database of previous design exemplars. These techniques analyse the given database and create a network that provides a sufficiently close representation of the stochastic phenomenon observed in the database. These algorithms use three main metrics to determine accuracy: *validity*, *understandability*, and *interestingness* (Mitra and Pal 2002). Validity measures what proportion of the data can be covered by the model. Understandability provides a complexity measure that can represent how easy it is for a designer to understand a model. Finally, interestingness measures the novelty of representation of a model in a design domain. These metrics have been listed in order of difficulty of measuring. Validity can be measured directly against the database supplied. Understandability requires a measure of human ability to understand a given model. Interestingness must be measured against the current state of domain knowledge and combined with a subjective element supplied by the domain expert.

4.1. INFORMATION CONTENT BASED METRIC

Most efficient BBN inducing algorithms require that the overall causal order is known prior to running the algorithm. However, where this ordering is not known, the complexity of most BBN graph inducing algorithm explodes to $O(n!)$, where n is the number of variables. In this research, it is assumed that the causal order of the design variables is not known prior to running the algorithm. A novel greedy algorithm has been developed for this work that reduces the computational complexity down to $O(n^2)$. This breadth-first greedy approach has been tested on some well known databases and performs well in terms of identifying the correct BBN. The overall process is illustrated in Figure 1.

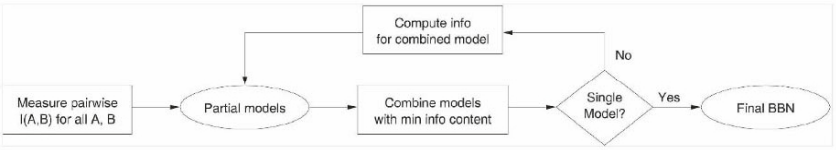


Figure 1. Flowchart representing the greedy BBN learning algorithm.

The graph search algorithm implements a greedy search heuristic based on a measure of the information content of the conditional probability distribution. Recall the definition of conditional probability:

$$P(B = b \mid A = a) = \frac{P(B = b, A = a)}{P(A = a)} \quad (1)$$

Where the events *A* and B are independent, P(*B*, *A*) = P(*B*)P(*A*). Hence, when *A* and *B* are independent P(*B*|*A*) = P(*B*). By considering the difference between the *observed* conditional and prior probability distributions, it is possible to measure the mean variation in this difference:

$$I(A,B) = E[P(B \mid A) - P(B)]^2 \quad (2)$$

The variation, *I*, represents how much more information is contained in the conditional probability distribution above the information contained in the prior probability distribution. A large value for *I* indicates that the conditional probability distribution contributes greatly to the knowledge of the domain while a small value indicates that the two variables are likely to be relatively independent of each other.

The graphical model search algorithm begins by measuring the pairwise information content between each variable pair. This is computed for both directions as in general *I*(*A*, *B*) ≠ *I*(*B*, *A*). For each design variable, the system is seeded with a partial model containing the given variable and the variable that has the greatest information content of its conditional probability distribution. Where a partial model would be repeated, the variable with the next highest information content is selected.

These partial models are ordered in increasing information content order. The next step is to merge partial models with low information content, creating a new partial model whose information content is given by the sum of its parts. The two lowest information content scoring models with a common variable are merged, resulting in one fewer partial models. Where there are more than two candidate models for combining, the tie breaker is determined by (1) resulting model complexity followed by (2) lower information score. This is repeated until all partial models are exhausted. The above greedy algorithm results in a single graphical model.

5. Implementation

To test the above design heuristics, it was necessary to implement the stochastic algorithm. To ensure wide access to the algorithm, it was decided to implement the interactive design support tool using Microsoft's Visual Basic (VB) within Excel. Most office desktops have access to Excel, and thus a large population of potential beta-testers exists.

The code is structured in two parts: The first part is a one-shot machine learning algorithm that uses Equation 2 to induce the network from a given dataset of prior design exemplars. As this only needs to be run once, it was written in Matlab rather than VB. While this restricts the ability for arbitrary

users to use their own dataset, this is not a part of the user trial. The second part of the code represents the user interface to the BBN. Figure 2 contains the flowchart for the iterative and designer led search process. This is encoded as a VB macro that reads the current design state from the Excel design spreadsheet and computes the PDFs of the unspecified design variables. These PDFs are extracted from the database of design exemplars that resides on a separate spreadsheet. The conditional PDFs are computed from the joint probabilities that can be extracted by frequency counting within the database. The remainder of this section will focus on the user interface.

Figure 2. Flowchart representing the overall design search process.

5.1. DATA STRUCTURE

The data structures for the interactive design search tool are based on the simple native structures available within Excel. There are three types of data that need storing: (1) the database of previous design exemplars; (2) the network structure; and (3) the current design state. Each of these is held in a separate Excel worksheet. While this is not a highly efficient approach, it does provide a very simple representation that can be easily manipulated by a designer. Typically, a designer would be only interested in the design status worksheet. However, the designer also has the capacity to edit the BBN directly in the case that it is believed to be inaccurate. Also, the designer is able to edit the exemplar database, either by removing data points or adding further ones. However, if the manually edited data had an impact on the network, this would not be possible for the user to determine directly.

The design status worksheet lists each design variable on a separate row. The first column contains the variable name. In the next column, the variable value is placed, when known. The remaining columns are used to display the PDF for the given variable. The PDF is computed for all possible values the design variable can take. This is a simple task, as the all the design variables have been discretised and so there are only a small number of values to consider. The designer then uses the PDFs as a guide to determining the next design variable value.

Similarly to the design status worksheet, each row of the network worksheet contains the network data for a single variable. The first column contains the variable name. The remaining columns contain the immediate causal 'parents' of the variable. For each variable, X, these represent the set of variables that X is causally dependent on. This set of parent variables is

typically denoted $\pi(X)$. Hence, in the BBN, the CPDF of X is expressed by $P(X \mid \pi(X))$.

Finally, the dataset work sheet simply contains a set of previous exemplar designs. Each design is listed on a separate row. The columns in this case contain the different design variables.

5.2. INTERACTIVE ALGORITHM

The interaction between designer and the code is centered around the unspecified design variables. For illustration purposes, denote the unspecified design variable as Y. To provide direct guidance, the information supplied for each unspecified design variable is reduced to a single dimension, namely the PDF for that design variable. Depending on the status of adjacent design variables, there are two main cases to be considered: (1) Y is a non-terminal node in the BBN tree and (2) Y is a terminal node. The BBNs that are induced from the greedy learning algorithm are tree structures: no node has more than one child, or alternatively, any variable can causaly only affect one other variable. However, a variable can have several parent variables that have a causal effect on it.

The first case is straightforward. The aim here is to compute the CPDF defined by $P(Y = y \mid \pi(Y))$ for all y values that the design variable Y takes. The CPDF only uses the specified parent design variables. That is, if one of the members of $\pi(Y)$ has not been specified, it is excluded from consideration. Clearly, if none of the parents have been specified, then the CPDF reduces to the PDF of the design variable Y.

In the second case, where the unspecified design variable Y is a terminal node, the code considers the child node of Y. As the BBN is a tree graph, there is only one child of Y. Let $X = \pi^{-1}(Y)$ be the unique child of Y. The designer is then presented with the following distribution:

$$P(X \mid Y = y, \pi(X)) \tag{3}$$

There are now two further sub-cases to consider: X has been specified and X has not been specified. Where X is known, the algorithm proceeds to compute the probabilities of achieving this specified value for all possible values $Y = y$ that the unspecified design variable can take. Again, only the known values of $\pi(X)$ are considered. In the second case, where X has not been specified, the only information that can be used to guide the designer is the PDF of the unspecified variable Y. This is as Y is a terminal variable, so there are no further parents that will affect it, and it is independent to the other parents of X, namely $\pi(X)$.

It should be noted that in this second case, Equation 3 is not a proper PDF as it does not necessarily sum to 1. This function measures the likelihood of achieving the already determined value of X. However, for the

purposes of identifying a good value for Y, the same argument applies, namely that a designer should focus on those values that provide a suitably high probability for achieving X's value.

All the PDFs are computed dynamically at run time by counting suitable exemplars from the database. The complexity of this process is $O(Nn)$, where N is the size of the database and n is the dimensionality of the design space.

5.3. DESIGNER HEURISTICS

The final aspect to be considered is how the displayed PDFs are interpreted by the designer as heuristics for the design search process. For each unspecified design variable, the relevant PDF for that variable is displayed in the columns adjacent to the design specification. As argued earlier, it is suggested that the designer focuses first on the variables with narrow distributions and then moves onto variables with ever wider distributions. This is the variable ordering heuristic. The second heuristic guides the designer to the value that each variable should be set to. It is suggested that the designer selects the value that has an acceptably high probability associated with it. This represents the most likely outcome for the design, or conversely, the design with the greatest likelihood of success.

6. Case Study: Preliminary Car Design

As an initial trial of the stochastic design search method, the well known UCI machine learning car design database was used (Blake and Merz 1998). This database contains a sample of 1728 designs, each with a full set of observations. Each sample represents a conceptual car design. The cars are represented as a 10-dimensional vector comprising of both design parameters and design characteristics. The design parameters are: the target purchase price; the expected maintenance cost; the designed safety level; the number of doors; the number of passengers; and the volume of luggage that can be carried. The design characteristics are: the overall cost of ownership; the comfort level; the technology level; and the overall car acceptability. All the design variables are discrete. A set of predetermined rules was used to map the design parameters onto the design characteristics to create the database that was then used by the greedy BBN induction algorithm. The structure of these rules is given in Figure 3. These structured rules provide a means for comparing the stochastic design tool to the original and defining structure of the design space.

The car database was first loaded into Matlab and passed to the BBN learning algorithm. This generated a network representing the causal links between the design variables. The algorithm produces exactly as many arcs as there are design variables. This resulted in a non-tree structure. In a tree

structure each node, with the exception of the root node, should have a single child. The structure that was produced by the learning algorithm had the 'safety' node linked to both the 'technology' and 'car acceptability' nodes. By considering the information content of the two arcs coming out of the safety node, the arc with the lower information content was deleted. The resulting tree network that was learnt from the dataset had an identical causal structure to the underlying rule structure used to create original the design database, as illustrated in Figure 3. This network was then encoded in the Excel spreadsheet, along with the database.

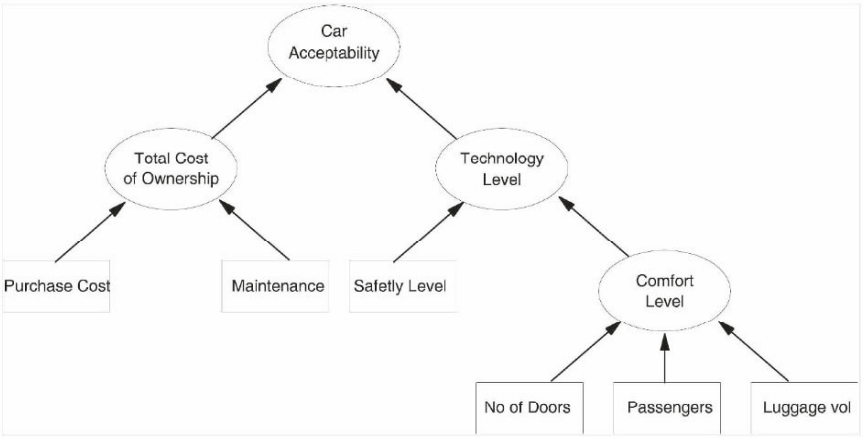


Figure 3. Rule structure for the conceptual car domain.

6.1. STOCHASTIC SEARCH

The Excel spreadsheet provides the 'user interface' to the stochastic design tool. Using this tool, four different design scenarios were explored based on the nature of the design specification: (1) only design parameters specified; (2) only design characteristics specified; (3) both specified; and (4) an 'infeasible' design specified. These are expanded below.

6.1.1. Design Parameters: 'People carrier'

In the first scenario only design parameters (design variables under direct control of the designer) were specified. Specifically, a subset of the design parameters were specified to reflect a partial set of the requirements of a 'people carrier' type car. The design specification required that the car should have low maintenance costs, a high safety rating, seat a large number of passengers, and have a large luggage space. This specification omitted the design parameters describing the purchase price of the car and the number of doors.

This specification was entered into the spreadsheet, and the VB macro computed the PDFs for the unspecified design variables. Figure 4 is a screen shot from this step. The stochastic design heuristic suggests considering the design variables with the smallest distribution first. Further, to maximise the likelihood of the design, the heuristic suggests selecting the values that maximise this PDF. In this case, the order and settings of the design variables were guided as follows (see also Table 1):

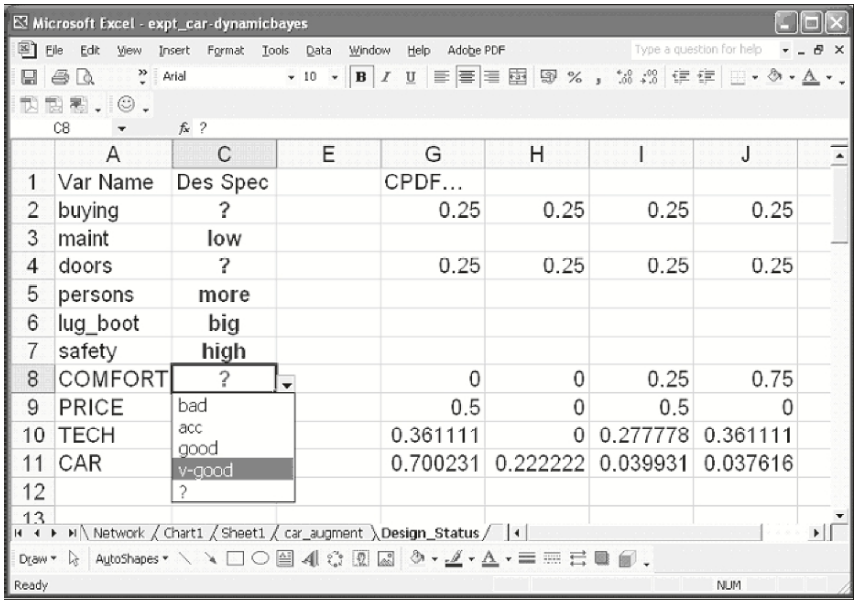


Figure 4. Screen shot from the 'People Carrier' design specification and initial PDF computation.

1. Comfort: set to 'high'
2. Technology: set to 'high'
3. Price: set to 'low'
4. Car acceptability: set to 'high'
5. Purchase price: set to 'low'
6. Doors: set to '5'

It must be noted that at each step there were other potential alternatives that could have been selected. Further, after each step, the PDFs for the remaining undefined variables did change, thus illustrating the dynamic nature of this search tool.

The final design does reflect a highly acceptable 'people carrier' design concept. This would then be taken through to a more detailed design phase.

6.1.2. Design Characteristics: 'Sports car'

The 'sports car' design scenario only specified the desired characteristics of the final design concept. Three design characteristics were specified: the car only required relatively low comfort, a high technology level, and a high overall ownership cost. This left a large number of design variables to be specified, which could potentially lead to infeasible designs without any guidance, Table 1.

TABLE 1. Search path for the unspecified design variables for the 'People Carrier'. Selected variable/value in bold.

Step	Variable	PDF/Likelihood			
1	Buying	0.5	0.25	0.25	0.25
	doors	0.25	0.25	0.25	0.25
	COMFORT	0	0	0.25	**0.75**
	PRICE	0.5	0	0.5	0
	TECH	0.6	0	0.28	0.36
	CAR	0.70	0.22	0.04	0.04
2	buying	0.25	0.25	0.25	0.25
	doors	0	1	1	1
	PRICE	**0.5**	0	0.5	0
	TECH	0	0	0	**1**
	CAR	0.70	0.22	0.04	0.04
3	Buying	1	1	0	0
	doors	0	1	1	1
	CAR	0	0	0	**1**
4	**buying**	**1**	1	0	0
	doors	0	1	1	1
5	**doors**	0	1	1	**1**

In a similar process to the previous scenario, the design heuristics suggested the following course of action, see also Table 2:

1. Safety: set to 'high'
2. Car acceptability: set to 'low'
3. Luggage space: set to 'low'
4. Purchase price: set to 'high'

In this scenario, there were two occasions where the shape of two PDFs were identical, Table 2, step 3, thus not providing a clear precedence for

determining the values. In these cases, it is for the designer to use their discretion to the order of determining the values.

The final design, while appearing to score poorly on a number of characteristics, is in line with a high performance sports car that has traded off mass appeal against a niche market.

TABLE 2. Search path for the unspecified design variables for the 'Sports car'. Selected variable/value in bold.

Step	Variable	PDF/Likelihood			
1	buying	0	0	0.25	0.5
	maint	0	0	0.25	0.5
	doors	0.33	0.33	0.22	0.22
	persons	0	0.5	0.33	
	luggage	0.58	0.25	0	
	safety	0	0	**1**	
	CAR	1	0	0	0
2	buying	0	0	0.25	0.5
	maint	0	0	0.25	0.5
	doors	0.33	0.33	0.22	0.22
	persons	0	0.5	0.33	
	luggage	**0.58**	0.25	0	
	CAR	**1**	0	0	0
3	**buying**	0	0	0.25	**0.5**
	maint	0	0	0.25	0.5
	doors	0.33	0.67	0.67	0.67
	persons	0	1	0.75	
4	**maint**	0	0	0	**1**
	doors	0.33	0.67	0.67	0.67
	persons	0	1	0.75	
5	**doors**	**0.33**	0.67	0.67	0.67
	persons	0	1	0.75	
6	**persons**	0	**1**	0	

6.1.3. Design Parameters and Characteristics: 'Accessible luxury'

The 'accessible luxury' design scenario specified a combination of design parameters and characteristics. The specified design parameters were: the car should have low maintenance costs; be a four-door design; and have a high

safety level. The car was to have the following characteristics: it should have a high comfort level and it should have a high acceptability level.

The stochastic search method suggested the following course of action, Table 3:

1. Technology level: set to 'very high'
2. Luggage space: set to 'high'
3. Overall cost of ownership: set to 'low'
4. Passengers: set to '4'
5. Purchase price: set to 'low'

In this scenario there were occasions where the guidance to selecting the variable value was ambiguous. For example, determining the overall cost of ownership placed equal weight between selecting 'low' or 'high' (see Step 3 in Table 3). In this case, as the car is intended to be 'accessible', the designer selects 'low'. Had the designer selected 'high', this changes the options that are offered two steps later when selecting the purchase price where the designer is offered 'high' or 'very high'.

TABLE 3. Search path for the unspecified design variables for the 'Accessible luxury'. Selected variable/value in bold.

Step	Variable	PDF/Likelihood			
1	buying	0.25	0.25	0.25	0.25
	persons	0	0.33	0.67	
	luggage	0	0.33	0.67	0
	PRICE	0.5	0	0.5	**1**
	TECH	0	0	0	
2	persons	0	0.33	0.67	0
	luggage	0	0.33	**0.67**	
	PRICE	0.5	0	0.5	
3	buying	0.25	0.25	0.25	0.25
	persons	0	1	1	0
	PRICE	**0.5**	0	0.5	
4	buying	1	1	0	
	persons	0	**1**	1	0
5	**buying**	**1**	1	0	0

6.1.4. Infeasible Design

In the final scenario, an infeasible design was specified. The design was determined to be infeasible according to the rules that map the design parameters onto the design characteristics. Specifically, given sufficient design parameter information to determine the value of a characteristic, the

characteristic was set to a different value thus representing an 'infeasible' design. While this is a slightly artificial case, it serves to demonstrate how the search method proceeds under such circumstances.

In this scenario, the stochastic search method reported a flat zero PDF for the cost of ownership characteristic (see Step 1 in Table 4). This indicates that under the current specification, there is no previous knowledge on what the likely cost of ownership for this design will be. If not having this information was acceptable, the designer could proceed with the current specification and find out further downstream in the design process what the value of this characteristic would be. As part of this case study, this option is not available. The alternative is to modify some other aspect of the design until a non-zero PDF arises.

To search for an alternative design specification that provides a non-zero PDF, the design domain BBN is used to track the parent and child variables of the cost of ownership variable. These are the purchase price, the maintenance cost, and the overall car acceptability. The design specification did not include either the purchase price or the maintenance cost, leaving the overall car acceptability design variable as the cause of the zero PDF. This leaves the designer with two options: either modify the child variable (i.e. the car acceptability) or consider the other parents of the child (in this case the technology level, Figure 3). This is as the PDF displayed for the overall car ownership variable is actually the CPDF of its child node, ranging over all possible values that car ownership can take. As such, the displayed PDF is the likelihood function, however the same value selection heuristics apply.

In this scenario, the designer decides to modify the comfort level variable. The designer slackens the specification on this variable until the PDF for ownership cost becomes non-zero, indicating that the design specification is feasible, Step 2, Table 4. Once the partial specification is feasible (no constant zero PDFs), the design search process continues as in the other (feasible) design scenarios.

6.2. NOTES ON TRADITIONAL SEARCH

A traditional approach to completing the design specification would in the first instance need to consider the design parameters and characteristics separately. While specifying the design parameters remains possible, as this is done directly by the designer, no information is made immediately available regarding the likely values the design characteristics would take on. These design characteristic values are only to be obtained if the designer has knowledge about the relationship between the design parameters and the characteristics. Without this knowledge, a designer must determine all design parameters and then obtain the design characteristics through more costly detail analysis or prototyping.

TABLE 4. Search path for the unspecified design variables for the 'Infeasible design'. The first step involves slackening the 'Comfort' variable. Selected variable/value in bold.

Step	Variable	PDF/Likelihood			
1	buying	0.25	0.25	0.25	0.25
	maint	0.25	0.25	0.25	0.25
	PRICE	0	0	0	0
	(COMFORT@v-high)				
2	buying	0.25	0.25	0.25	0.25
	maint	0.25	0.25	0.25	0.25
	PRICE	**1**	0	0	0
	(COMFORT@high)				
3	**buying** maint	0.5	**0.25**	0	0
		0.5	0.25	0	0
4	**maint**	**1**	0	0	0

The reverse approach where the designer specifies the design characteristics and then searches for appropriate design parameters is not directly possible with a traditional search. Where no or little knowledge exists, the designer must guess initial design parameter settings and then test. This must be repeated until either a sufficiently good design is achieved or enough knowledge is generated to be able to understand the design domain sufficiently well for the purposes of meeting the specification.

Both these approaches require performing extensive number of experiments where the designer lacks knowledge on the nature of the relationships between the various design variable.

7. Discussion

There are two aspects to this stochastic design search method: inducing the BBN design model from previous design exemplars and using the BBN as a search tool. The information based induction algorithm appears to perform well, based on a series of tests using databases taken from known source models. The car design database provided an example of this, where it identified the network structure with a single extra arc. This spurious arc was easy to identify, as it was the arc with less information from one of two potential arcs that broke the tree structure.

Using the BBN induced from the design database as a dynamic search tool offers an efficient search strategy when the two search heuristics are employed. The feasible design scenarios mainly followed the search

heuristics, with the designer rarely 'deviating' from the first ranked choice. Further trials are needed where the designer does not follow these suggestions.

Where a designer starts with an infeasible design, as per the final design scenario, the stochastic search tool simply reports constant zero PDFs for the unspecified variables. In the reported scenario, the designer used knowledge of the BBN structure to identify the 'neighbouring' design variables to modify blindly. An improvement would be to provide some form of guidance to identify fruitful modifications to the current partial design specification. This would allow the designer to 'unblock' the infeasible design specification using a minimal change to the original specification.

8. Conclusions and Future Work

Using the Bayesian Belief Network with the two search heuristics provides an efficient conceptual design search tool. The two heuristics aid the designer to first identify the next design variable that should be determined, followed by which value would provide the most robust design. A powerful aspect of the BBN approach is that the designer need not distinguish design parameters from design characteristics. This allows a designer to specify design characteristics that are not normally under a designer's direct control. However, it must be emphasised that the designer is not constrained by the design heuristics and is free to explore the design space in other orders. This offers the designer the flexibility that is essential during the conceptual design stage.

Further work is required in a number of areas. Research is needed on how to develop a more intuitive user interface to the BBN. There is a need for metrics for PDF 'spikiness' versus 'flatness'. This is critical as it will not be possible for a designer to identify the narrowest of PDFs in a design domain with considerably more variables. Another key area for further work is to develop methods for identifying design variables in infeasible design specifications that could be fruitfully slackened. Currently, the designer only has the network to identify neighbouring variables but no information on which variable should be modified.

Finally, this work was based on an artificial database with a fully tested set of designs (in terms of the design parameters). Further investigations are required where this is not the case, as this represents real design situations.

Acknowledgements

This research is funded by a Nuffield Foundation Award to Newly Appointed Lecturers in Science, Engineering and Mathematics (Grant number: NAL/00846/G).

References

Blake, CL and Merz, CJ: 1998, UCI Repository of machine learning databases, Available Online: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Clarkson, PJ and Hamilton, JR: 2000, Signposting: a parameter-driven task-based model of the design process, *Research in Engineering Design* **12**(1): 18-38.

Jensen, FV: 2001, *Bayesian Networks and Decision Graphs, Statistics for Engineering and Information Science*, Springer, NY.

Mitra, S and Pal, SK: 2002, Data mining in soft computing framework: A survey, *IEEE Transactions on Neural Networks* **13**(1): 3-14.

Pahl, G and Beitz, W: 1996, *Engineering Design: A Systematic Approach*, Springer-Verlag London, 2nd ed.

Pugh, S: 1990, *Total Design*, Addison-Wesley.

Taguchi, G., Elsayed, EA and Hsiang, TC: 1989, *Quality Engineering in Production Systems*, McGraw-Hill.

Wallace, DR, Jakiela, MJ and Flowers, WC: 1996, Design search under probabilistic specification using genetic algorithms, *Computer Aided Design* **28**(5): 405-421.

Ziv-Av, A and Reich, Y: 2005, SOS — subjective objective system for generating optimal product concepts, *Design Studies* **26**(5): 509–533.

COGNITIVE STUDIES OF DESIGNERS

A comparative study of problem framing in multiple scenarios
Thomas Kvan and Song Gao

Comparing entropy measures of idea links in design protocols
Jeff Kan, Zafer Bilda and John Gero

Analysing the emotive effectiveness of rendering styles
Raji Tenneti and Alex Duffy

Impact of collaborative virtual environments on design behaviour
Mary Lou Maher, Zafer Bilda and Leman Figen Gül

A COMPARATIVE STUDY OF PROBLEM FRAMING IN MULTIPLE SETTINGS

THOMAS KVAN
University of Sydney, Australia

and

SONG GAO
Peking University, Hong Kong

Abstract. Problem framing is an essential element of the design process because it is an important design activity in solving design problems. It is the first part of a cyclical design process which involves "framing", "moving", and "reflecting". Framing activities can be considered as a typical cognitive design process involving several levels. As an essential design activity, framing can be considered an indicator to trace whether digital media changes the way designers engage in their work. The results indicate that problem framing activities are significantly different in an online remote setting as compared to the two other settings. It appears that a chat line-based remote setting does not only facilitate a greater proportion of framing activities, particularly high level framing, but also shows more richly interlinked design activities.

1. Introduction

Studies on categorization, feature, and relation of different kinds of design activities have been carried out since 1960. Eventually, different design paradigms were developed, in which design activities with different names were found to be related with each other under particular relationships identified in each paradigm. The investigators of this study chose problem framing as a particular case to explore because this design behavior occurs in each design paradigm and is an essential part of the design process. Problem framing was chosen as the indicator to trace whether digital media changes the way designers engage in their work as compared to paper media.

J.S. Gero (ed.), Design Computing and Cognition '06, 245–263.
© 2006 *Springer. Printed in the Netherlands.*

2. Problem Framing

Design problems have been categorized as well-defined, ill-defined or wicked, reflecting the extent to which their solutions are immediately apparent. Design processes have been explored extensively and described according to features of these different categories of design problems. In particular, Schon's cyclical design process, which he describes as a "reflective conversation with the material of the situation", has been extensively used in research into design education and design activities. Schon has referred to the act of problem definition as "problem framing", the term we will use in this paper. He postulated that the activity of framing was central to a successful process of design and hence is a key activity in design.

2.1. MEANING OF PROBLEM FRAMING

When a designer faces a design problem, problem framing is the initial step taken. "It requires first the discovery of a problem area or topic, and later the structuring of the problem into a workable springboard for solution generation"(Jay and Perkins 1997). Reitman (1964) claimed that problem transformation plays a key role in solving ill-defined problems. Within this transformation, each subproblem (or which he called as a problem vector) as an information structure might be considered as a plan involving a state, process, or objects. Simon (1984) emphasized the importance of the planning method in dealing with ill-defined problems. "Planning was done by abstracting from the detail of a problem space, and carrying out preliminary problem solving in the abstracted space". This "abstracting" aims to decompose an ill-defined design problem into a smaller problem. Problem framing is a key element adopted to transform an ill-defined or wicked problem into a well-defined one. On the other hand, designers, are not limited to "given" problems, but they instead find and formulate problems within the broad context of the design brief (Cross 2001). Through the formulation and reformulation of design problems, planning can be constructed to imagine; a scenario can be documented involving setting goals and following rules (Coyne 2005).

2.2. COGNITIVE PROCESS

Problem framing is a developmental process involving systematic transformation. It is not just an external design activity represented in a variety of design media, but is also a process influenced by human memory and the outside environments. According to Schon's description of problem framing, we can observe several categories of those actions. In his words, "As [inquirers] frame the problem of the situation, they determine the

features to which they will attend, the order they will attempt to impose on the situation, the directions in which they will try to change it. In this process, they identify both the ends to be sought and the means to be employed"(Schon 1983:50-54). Here, three different categories of problem framing can be identified, namely, conjecture, setting rules, and planning. Framing activities, therefore, play different roles in the design process. From Minsky's study, problem framing could be classified into different types or levels. This classification is a common phenomenon in some cognitive and social studies. This section will analyze some studies on the categorization of problem framing. As mentioned earlier, problem framing can be considered as a process of searching and transformation. A similar classification of human thinking has appeared in other studies, showing a strong cognitive background.

3. Assumption About Design Tools

Paper-based design tools have been dominant in studio teaching since the formalization of design learning in Ecole des Beaux Arts. For many designers, the medium is inextricably bound into the activity of designing, with designers largely using paper to frame design problems (Robbins and Cullinan 1994). Indeed, the advent of the use of paper in designing has been noted as the moment at which design became an intellectual activity (Wigley 2001). As compared to digital design tools, paper seems to afford more predictability, and therefore, greater communication (Sellen and Harper 2001). Others suggest that digital design tools inhibit communication between the mind and the hand because of the precision demanded by the system, hence interrupting the conversation of design and disrupting framing activities (Lawson 1994; Corona Martinez and Quantrill 2003). Corona-Martinez and Quantrill observed that a computer is not a drawing instrument like a pencil, but it engages the designer in a different relationship with the act of drawing, changing the act with "an intermediate system of drawing according to our indications provided by the pressure on the button of mouse, which in turn responds to the feedback from our sight of what appears on a screen … something new has invaded the apparently intangible craftsmanship of drawing" (Corona Martinez and Quantrill 2003). Interviewing several architects, Lawson (1994) found that most of them preferred using paper-based design tools to help them in design thinking, and then using digital tools for documentation and presentation rather than as part of the design process. Burton, for example, considered that when interacting with a computer, he feels difficult to modify a drawing directly. "This close interaction between himself and his drawing leaves Richard Burton personally unenthusiastic about the idea of computer-aided design, of which he makes no use himself. He considers that the directness with which

he can alter a drawing is missing when mediated by a computer, and thus the feeling is lost". Similarly, Wilford preferred paper and pencil in designing. Echoing Schon's 'conversation', he called this process of designing an iterative and comparative process, claiming that it is impossible for designers to be detached from "this very immediate process of drawing lines on paper and tracing through".

Several studies have been carried out to investigate the effects of computer collaborative tools on design communication. Kvan et al. (1997) found that designers engage in more high-level communication when using textual communication tools as compared to using video conferencing tools. Gabriel and Maher (1999) conducted three sets of experiments adopting different types of communication, namely, face-to-face communication, computer-mediated communication using video conferencing, and computer-mediated communication using a chat line. They, too, found that design communication was significantly different among the three settings, and in particular, that text better supported design collaboration. Neither these nor other studies have identified the ways in which the media supported problem exploration and definition, noting only that problem spaces were more widely searched in textual-based communication.

4. Research Method

In this study we compare the pattern of problem framing activities in paper-based to digital-based settings. Protocol analysis is selected as research method to explore this design activity. Statistical analysis and linkograph study are employed to analysis data

4.1. PROTOCOL ANALYSIS IN DESIGNS STUDIES

Protocol analysis may not be explicit enough in studying design inquiry, but it is a more solid and thorough method as compared to other examinational techniques. It has been adopted in different design disciplines like mechanical engineering, software design, electrical design, industrial design, architecture, and interior design. As a popularly used method, it brings out into the open the mysterious cognitive abilities of designers (Cross 2001), and is well suited for the comparison of what we are interested in (Goldschmidt 1995). This method includes soft and hard techniques. The soft technique refers to the observation of design activities, and the hard technique refers to coding and analysis mechanisms (Oxman 1995). Gero and Mc Neill (1998) conducted protocol analysis to explore designers' intention and described this method in great detail. One distinct characteristic of this method is that when subjects conduct the task, they are required to concurrently verbalize what they think. Concurrent verbalization

is considered as an equivalent of the cognitive process in humans (Ericsson and Simon 1993).

In design studies, protocol analysis was first adopted by Eastman to study design cognition (Eastman 1968). Starting from Eastman's first usage of this method, protocol analysis has four types which are "think aloud", "retrospective analysis", "teamwork analysis", and "replication protocol analysis". Teamwork protocol analysis is developed from the concurrent verbalization analysis. This method was first adopted in the workshop of Delft University of Technology (Dorst 1995). Since then, this method has been adopted to test the effects of computer-supported communication tools on the design process (Goldschmidt and Weil 1998; Kvan et al. 1999; Gabriel and Maher 2000). Teamwork protocol analysis could solve to a certain extent the problems previously mentioned because the discussion of design issue is not only a regular activity in the design process (Cuff 1991; Kvan 2001), but also expresses the social and perception aspects of design (Cross and Cross 1995; Dorst 1995). Instead of requiring individual designers to think aloud, it allows two or more designers to work together. Through recording their discussion, the design protocol can be naturally elicited (Goldschmidt and Weil 1998).

4.2. DESIGN EXPERIMENTS

This study aims to identify the difference in problem framing in three design settings. To explore this design activity, the subjects are required to use computer tools or paper-based tools to solve a simple wicked design problem within a given time. Eighteen pairs of students are the participants of the design exercise. They are equally assigned into three settings which are online remote setting, online co-located setting, and paper-based co-located setting. In both co-located settings, the verbal and visual design protocols are videotaped. In online remote setting, the textual and visual design protocols are recorded by a computer. After transcribing the verbal design protocol, two types of coding scheme are adopted to examine the problem framing activities. The techniques adopted to measure problem framing are statistical analysis and linkographic study. Inter-coder reliability is conducted to validate this measurement.

4.3. CODING SCHEMA

The coding schema is based on Schon's "framing", "moving", "reflecting" design process and has been described in detail elsewhere. This model is employed to isolate framing from design process. The description of coding schema adopted is shown below. Table 1 shows the definition and correspondence examples of Schon's model.

4.4. LINKOGRAPHY STUDIES

Protocol data are loosely organized and analyzed; linkography study holistically describes protocol data in an organized way. The first of part study identifies there are different ways designer's engage in problem framing under different design environments. Through employing linkography technique, the whole structure of design activities can be unfolded, from which we wish to discover whether framing activities show any difference within the structure of linkography. In this study therefore we not only use statistical method for data analysis, but also adopt linkography technique to structure links among problem framing with other activities within different contexts. Using the protocols in which we have previously encoded designs actions using the Framing-Moving-Reflecting model, the linkograph reveals the interconnected actions and thus the depth of an idea. Framing, as one type of design moves, is more specific. We define several terms generated from this system. A component is the unit in which all design moves are inter-linked. A diameter means the number of the nodes linking two design moves in one component, thus the greater the diameter, the larger the component. We isolate the largest component from each setting, and adopt statistical technique to compare the incidences of problem framing in three designed settings. By this technique, the pattern of a design process can be visualized holistically.

TABLE 1. Coding schema and examples (after Schon).

Coding category	**Definition**	**Examples**
Framing	Identify a new design problem; Interpret further from design brief;	"We have to provide a sense of arrival at each site access point."
Moving	Proposed explanation of problem solving, a tentative solution.	"Maybe some here can put the playground"
Reflecting	Evaluate or judge the explanation in 'moving'.	"I think it is ok. Just represent the design"

Table 2 illustrates the definition of terms adopted in this paper. These terms are used to measure this triangular web. Some of them are same with the terms used by Goldschmidt; others derive from the domain of graph theory.

TABLE 2. The definition of terms derived from linkography.

Name	Abbr.	Description
Links	L	The number of linked design moves in a component; the larger the diameter of a single component, the more extensive a design thought.
Index	I	A process or a portion of it is the ratio between the number of links and the number of moves that form them (Goldschmidt 1990).
Component	C	One unit in which all design moves are inter-linked; the larger the number of components, the more fragmented the design session.
Diameter	Di	The number of linked design moves in the largest component in one setting; the larger the diameter of a single component, the more extensive a design idea.
Depth	De	The largest number of nodes linking two discrete design actions in a component and hence describes complexity of relationships between design actions.

5. Results

By employing these method, two parts of results can be observed in the below, which are framing in Schon's design conversation and linkography study.

5.1. FRAMING IN DESIGN CONVERSATION

Schon's design model is largely rooted in studying the relationship between design activities with tools, and thus has been adopted in many human-tool interaction design studies. This model includes three separated design actions: framing, moving, and reflecting. The three design activities compose a cyclical design cycle. This theory was developed from a paper-based design setting in which Schon observed the discussion between a tutor and a student in a design studio. Through analyzing the 20-minute discussion, Schon claimed that the tutor used drawing and talking as the design language to help him engage in the design conversation. In this co-located setting, verbal discussion and paper-based sketch are the tools which the designer adopted to support his design communication. If the design tools are different, will this same cyclical design pattern occur?

5.1.1. Within the variable

Schon's design model demonstrates that framing is an initial activity evoking other design actions. We compare these design activities (including framing, moving, and reflecting) in different design environments. We find that when the modes of both design tools are different from those in the paper-based setting, there is a significant difference among the three design activities (remote setting); when the mode of the drawing is different, however, no significant difference is found when comparing the three design

activities. Table 3 presents the communication mode for each setting and the results of the ANOVA test. From Table 3, we can observe that in a remote setting, the F value is 7.61 larger than the critical F value. Meanwhile, the P value is 0.003 which shows a significant difference when comparing the three design activities, whereas in both co-located settings, no significant difference is found. These results indicate that a remote setting fundamentally changes the way designers engage in the design process as compared to both co-located settings.

5.1.2. Between the variable

We counted the number of these design activities in terms of this "framing-moving-reflecting" model, and calculated the percentage of the three categories. Figure 1 presents the number of the three design activities across the three settings, and Figure 2 shows the percentage of the three design activities across the three settings.

TABLE 3. Results of ANOVA Test in comparing three design activities in each setting.

The name of setting	Communication mode	Categories	F	P
Remote setting	Chat line and digital sketching	Framing vs. Moving vs. Reflecting	7.61	0.003
Digital co-located setting	Verbal communication and digital sketching	Framing vs. Moving vs. Reflecting	0.74	0.49
Paper-based co-located setting	Verbal communication and paper-based sketching	Framing vs. Moving vs. Reflecting	0.27	0.77

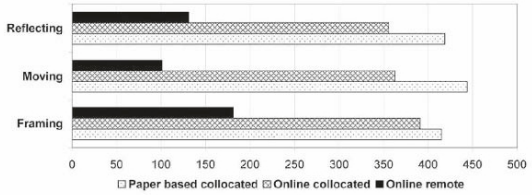


Figure 1. The number of the design activities across the three settings.

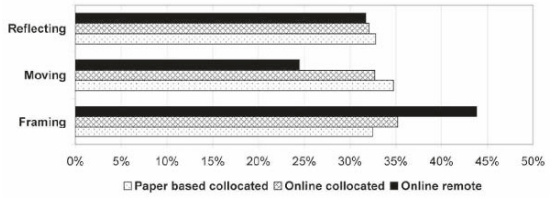


Figure 2. The percentage of the design activities across the three settings.

As indicated by the bar charts above, there is a substantial difference in the total communication in each setting. It can be seen that the number of framing activities is 415 in the paper based co-located setting, 391 in the online co-located setting, and 181 in the online remote setting. The number of communication in the other two categories is similar and proportionate. Thus, the results suggest that face-to-face working supports greater communication than digital-supported textual communication. In co-located environments, the subjects can draw while talking, but in a remote environment, the two actions are separated. When we calculate the percentage of framing in these settings, we find that the percentage of framing is higher in an online remote setting than in paper and online co-located settings (43.8% online remote, 32.5% paper-based co-located versus 35.2% online co-located setting). The chat line setting therefore appears to support a greater proportion of framing than the two other activities (moving and reflecting).

5.2. LINKOGRAPHIC STUDIES

Linkography study gives us a holistic way on studying framing by using a design graphic system. Three parts of study are incorporated. First studies full graphs of linkograph; secondly by quantifying linkographic representation, largest components from each setting are isolated for comparison. Lastly a case study is adopted to explore the effects of framing on latter activities.

5.2.1. Quantitative results

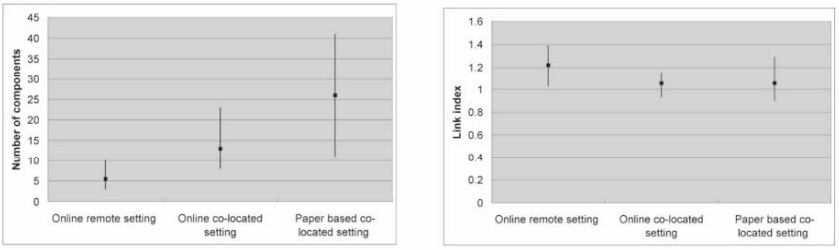
To identify developmental steps in designing, we have used the graphic technique of linkographs to track the development of design ideas. Developed by Goldschmidt (1990), the linkograph is a tool to encode design activities by identifying interlinked design moves by way of a systematic triangular web. We analyze the resulting graph using graph theory to distinguish the graphs.

To examine the implication further, we have represented the encoding using linkograph to examine the connectedness of frames, moves and reflection throughout the design sessions.

- Full graph

The following figure shows maximum, minimum, and median of number of components (Figure 3). The highest median of number component is in paper based setting; next is online co-located setting, last is online remote setting. We assume that the less number of components, the more connectivity of linkage among design activities. This figure therefore indicates that online remote setting has higher degree of connectivity of links.

The term of link index (LI) is adopted from Goldschmidt's study (1990). According to her, "Low L.I.'s were found in the cases of inexperienced designers and those experiencing difficulties in dealing with a particular design problem"(Goldschmidt 1990). Figure 4 presents the link index of full graph of each setting. The highest value of link index is in remote setting, next is paper based co-located setting, last is online co-located setting. The higher value of link index, the more design productivity; this figure indicates online remote setting (with chat-line based textual communication) could achieve richer links of design activities. The results of full graph suggest remote setting better support design productivity and show richer links than other two co-locates settings.



(a) (b)

Figure 3. (a) The number of components across the three settings; (b) the link index across the three settings.

Largest component:

Using Goldschmidt's (1990) index, the largest component (the component with largest number of diameter in each setting) in each protocol in setting was measured and the mean of these numbers calculated. From this measure, we see that the largest mean index number is obtained for the online co-located setting. The measure of the index, however, does not inform us of the breadth of the complexity in the design activity. To identify this, we have

employed the three standard graph descriptors introduced in the section above, component, diameter and depth.

Table 4 compares the mean value of total number of components and diameters across the three settings; and their ratio. For each metric we have shown the mean of each across the six protocols recorded in each setting. The first column shows the mean value of the number of component in each setting. The mean of the number of components in remote setting is much less (5.8) compared to paper (26.8) and digital based co-located settings (13.5). The next row presents the mean value of the number of links. By comparing the mean value of number of links among the three settings paper co-located setting contains largest number (238); remote setting has 81; and digital co-located setting has 188.We assume that if fewer components occur while richer links found, then leading to design complexity. In other words, the ratio of link and component (ML/MC) is an indication of the design productivity. The ratio in online co-located setting is the highest (14.90), next is online remote setting (13.96) and the last is paper co-located setting (8.88).

TABLE 4. Component and diameter among the three settings.

	Number of Component		**Number of Link**		**Ratio (ML/MC)**
	Mean (MC)	**SD**	**Mean (ML)**	**SD**	
Online remote setting	5.8	2.79	81	21.40	13.96
Paper co-located setting	26.8	10.76	238	49.86	8.88
Online co-located setting	13.5	5.5	188.8	74.01	14.90

Table 5 describes the index metric of the largest components among the three settings and three largest components in each setting. In each design session we choose the largest component, thus totally six components in each setting. The first three columns compare the mean value of total link; mean value of the numbers of moves; and index. Results indicate the mean value of total link and the mean value of numbers of moves in paper-based co-located setting is the highest (85.7; 121); next is the remote setting (59.7; 61); and in digital based co-located setting the mean value of total link and moves are 87 and 93.3. When comparing the value of index it shows digital co-located setting holds the largest number (1.12); remote setting is the next, which is 1.07; and paper-based co-located setting is 0.71.

By isolating the largest component in the three settings for investigating we find that digital co-located setting contains the largest number of diameter (164); the greatest depth is 5. The remote setting holds the largest number of greatest depth (9) though the diameter of it is less than that of

digital face-to-face environment that is 83. The largest component in paper-based setting has 132 diameter and 7 of the greatest depth.

As described above, a component is a unit of inter-connected design moves, which represents the process of the development of design ideas. We observe that remote setting has far fewer components, that is, far fewer discrete design threads are developed which then are abandoned and not continued; designers appear to engage in more limited exploration of a problem. In addition, the remote setting exhibits the greatest depth among components. From this, we observe that purposeful design activity is more often engaged in the remote setting. This suggests that initial ideas developed and recorded in the remote setting, where a chat line is employed, are more persistent while in the paper-based setting the idea is developed sequentially. Although online co-located setting has the largest number of diameter and the highest value of ratio, the depth is only five even less than paper-based co-located setting. This implies that the design ideas raised in online co-located setting are not re-visited or re-modified as often as those raised in other two settings.

TABLE 5. Index metric of the largest components among three settings.

	Total link		**The number of moves**		**Index**		**The largest one in each setting**	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**The Diameter (Depth)**	**Greatest Depth**
Online Remote setting	59.7	16.81	61	17.25	1.07	0.32	83 (9)	9 (83)
Paper-based co-located setting	85.7	36.45	121	25.20	0.72	0.32	132 (4)	7 (83)
Online based co-located setting	87	49.78	93.3	54.73	1.12	0.05	164 (4)	5 (147)

5.2.2. Descriptive case study

Previous studies focus on the dimension of largest components and greatest depth and ignore the relationship of largest components and greatest depth with full graphics. Here we introduce two analysis techniques. The first analysis technique is to calculate two sets of ratio values. The first is to calculate the ratio value between number of design activities of largest

component and the total number of design activities in each setting. The second is to calculate the ratio value between the greatest depth of linkograph and the total number of design activities in each setting. The second technique is case study of greatest depth in the three design environments. Through this case study we wish to understand how an idea was developed and improved.

The protocols recorded for each session were coded using the schema above to identify frame, move and reflection actions. Framing actions were then further encoded using the second schema to identify high and low level framing. Linkographs were drawn for each session.

Goldschmidt (1990) devised a measure of link index to measure the comparative complexity of critical moves and to identify the critical path in a linkograph. As she noted, a low index value suggests inexperience and a poor grasp of the design problem. The index does not tell us of the complexity of the links in a graph, so we have identified two additional metrics. One is the ratio value that tells us of the structure of the graph, and the second, depth, measures the continuity of design ideas across a graph.

Ratio value:

These protocols have been previously reported in Kvan and Gao (2005) where analysis was conducted on the largest component only in each setting. For these, the depth and diameter were calculated; in that analysis, we found that the largest components in the remote settings exhibited the largest depth but not the largest diameters. That analysis, however, did not examine the largest component in its larger context of the whole protocol. The analysis technique we adopt in this study emphasizes the relation between largest component and the whole design process. To examine this, we have used two ratio values:

$$\text{Ratio value 1} = \frac{\text{the number of design activities of largest component}}{\text{the total number of design activities in the protocol}}$$

This ratio value, R1, measures the coherence of the largest component within the overall design process. If the ratio is low, the largest component represents but a small part of the overall process and hence suggests that the overall process was fragmented into many discrete and disconnected design actions. A large R1 indicates a persistency in design ideas as they are re-examined and interpreted, in which instance the largest component represents a major part of the design process thus suggesting internal coherence and continuity.

$$\text{Ratio value 2} = \frac{\text{the greatest depth in a component for the protocol}}{\text{the total number of design activities in the protocol}}$$

Ratio value R2 measures the reach and extent of revisiting of early ideas as the design progresses. Depth is a measure of the largest number of nodes linking two discrete design actions in a component and hence describes

complexity of relationships between design actions. If a protocol exhibits a high R2, ideas are closely linked from beginning to end of the design session. A low R2 indicates rapid chaining of concepts but little cross-checking to earlier intentions. Using the terminology from Vygotskii's complexes, a high R2 suggests complexes of a higher order.

Results from calculating the two R values are presented for the three settings in Figure 5; in each setting the minimum, maximum and median are indicated. We can see that the value for R1 and R2 are substantially higher in the remote setting.

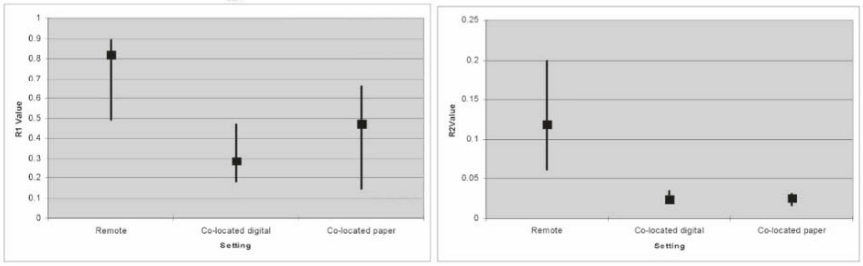


Figure 5. Results for R1 and R2.

Case study

As we have reported elsewhere, the proportions of framing found in these protocols varied significantly in the three settings (Figure 6), with the remote setting exhibiting a higher ratio of framing than the co-located settings and that the protocols from the remote setting exhibited more of both high and low level frames that the co-located settings.

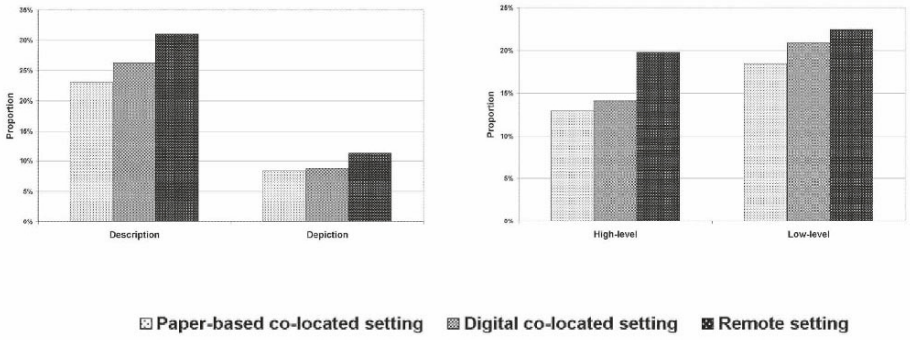


Figure 6. The comparison of the proportion of different frames of the largest components across the three settings (left: description vs. depiction; right: high and low-level of frames).

These initial findings are clarified by examination of three examples, one from each design environment, in which we can examine how a creative idea is introduced, improved and carried to later activities. Figures 6, 7 and 8 we

show the linkograph for the largest component with the highest R1 value for each of the settings.

Remote setting:
In this design environment, textual communication and digital drawing are main channels for design communication. The first protocol, Figure 7, shows designers' creative design ideas interlinking richly and over a considerable span of the design session. Design ideas are constructed into complexes by association and continuation of thoughts. We observed in this protocol that this pair of students continuously produces several high-level frames to correspond the first problem, like underground, open structure, evoked by the first high-level framing. Framing activities represent 39% of the actions in the component. The R2 value is 0.06, the lowest R2 of all remote protocols, indicating other protocols should yet richer complexes.

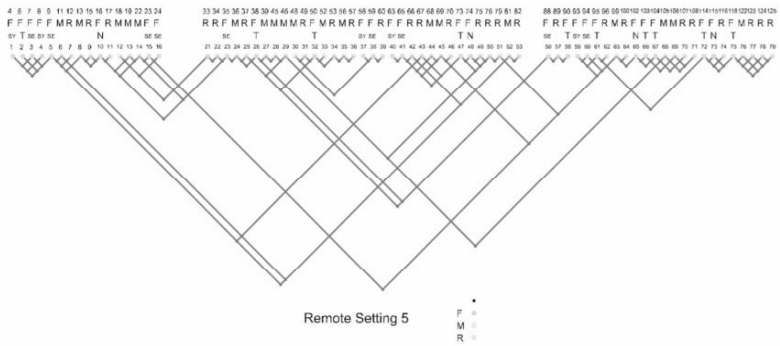


Figure 7. Highest R1 in the remote setting.

Digital co-located setting:
In this design environment, verbal communication and digital drawings are the primary channels adopted by designers for design communication. From the protocol, Figure 8, we observe a small amount of chaining even though this is the highest R1 recorded; framing occurs 29% of the time. The R2 value is 0.0234 which is close to the median value for this setting.

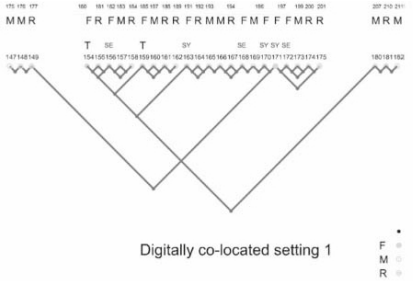


Figure 8. Highest R1 in the digital co-located setting.

Paper-based co-located setting:
In this design environment, verbal communication and paper based drawing are main channels adopted by designers. The design engagement in this component has a short span (action 2 to action 48) but the, Figure 9. Frames represent 28% of the actions; the R2 for this component is 0.0247, close to the median value for the setting.

Through comparing the content of each highest R1 of the three design environments, we find that in remote setting early designs idea lead to larger complexes with more frames than in both co-located settings. In this comparison writing communication again shows its advantage in problem framing process.

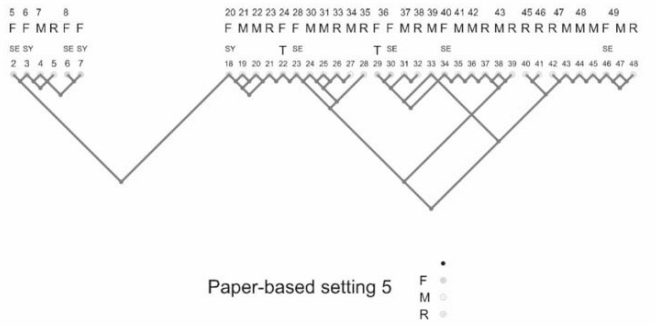


Figure 9. Highest R1 in the paper-based setting.

6. Discussion

Problem framing is closely related with Schon's design paradigm; therefore, it is necessary for us to investigate the relationship between design tools and the design process. Zeisel (1984) and Schon (1985) identified a design cycle designers engage in. The design process is like a spiral consisting of three consecutive design activities: framing, moving, and reflecting. According to the results, this design metaphor or design paradigm is well suited to co-located settings, while in a remote setting, it is changed by textual communication. This change confirms the theory of affordance (Sellen and Harper 2001) in which the properties of design tools determine the possibility of actions. Schon's design paradigm was developed from the observation of the conversation between a tutor and a student. In such situation, the two persons mainly adopted verbal communication and paper-based sketching as the design tools for the design process. In a verbal communication environment, the designers produce the problem to solve, and then propose a possible solution for it, After which, they test or evaluate this solution which might evoke other problems. However, in a remote

setting, this is not the case. We find that sometimes, designers propose several problems in advance, and in other times, they propose a serial of solutions. In a chat line-based remote setting, the design communications are stored in a computer which helps designers unload their design exchanges in an explicit way. This characteristic of a remote setting delays designers from referring to their previous design exchanges, thus prolonging this design to be spiral. Schon's design paradigm is a normal thinking process which professionals usually adopt in their career (Schon 1983). Sketch-supported verbal communication has its characteristics in supporting this thinking process (Schon 1983; Fish 1996). However, sketch-supported textual-based communication changes this thinking process to some extent, raising the issue on whether or not this design tool has its own potential in supporting particular design activities.

This study has introduced the use of linkographs to measure protocols in an effort to characterize the richness and complexity of design activity. We have applied three new measures, components, diameter, and depth, as the metrics of richness, and have adopted the idea of Goldschmidt's index. We have demonstrated that these measures correlate with the findings derived in earlier papers using statistical measures. The conclusion is that the discussions on design activity in chat lines can be measured to be richer in design complexity than those taking place face to face. Goldschmidt (1990) noted that "design productivity is related to the generation of a high proportion of design moves which are rich with links". In her paper, Goldschmidt developed an index as a measure of the percentage of linked moves. She proposed that a higher linkage value is an indicator of greater interconnectedness in design moves. In this figure, we observe that the pattern of design activities in the remote setting are richly interlinked as suggested by the considerable interconnections in the design activities; the framing activity in this setting appears to have an impact on subsequent moves. In a co-located setting, however, many design moves are isolated and disconnected. Thus, the remote setting seems to facilitate richer design productivity than co-located settings.

7. Conclusion

Preliminary study demonstrates that using digital design tools do not interrupt design process, suggesting that the preconception of digital tools interrupting the design conversation is unfounded, at least in so far as the conversation is measured as a framing process. The results also suggest that the design tools have some influence on the activity of problem framing. It would appear from this study that digital tools belong in the category of tools that contribute to the process of design and should not be relegated to a role of supporting 'hardlining' and presentation after design thinking is

completed. The positive effect observed here in the non-co-located setting continues to surprise the researchers as it is counter-intuitive and is not supported by the lore of designing, deserving further consideration. Overall, these experimental findings show that digital design tools appear *not* to be detrimental to problem framing activities and may also enhance the efficiency of design exploration, contrary to commentary from practitioners. If digital tools are constraining inventiveness and disrupting the design conversation, it may not be in the density of framing activities but other aspects of the design process that are being disrupted. Digital design tools therefore show some potential to enable exploration beyond traditional design processes, perhaps to fulfill the wish to free designers from the traditional paradigm of designers formalized in the era of Renaissance.

This study identifies that there is indeed significant difference of framing activities when using different design tools; especially in remote setting framing activities are proportionately higher. Linkograph study shows richer links in remote setting in terms of link index in full graph and ratio value. Future study could be employed to examine the reason why these incidences occur.

References

Corona Martinez, A and Quantrill, M: 2003, *The Architectural Project,* Texas A&M University Press, College Station.

Coyne, R: 2005, Wicked problems revisited, *Design Studies* **26** (1): 5-17.

Cross, AC and Cross, N: 1995, Observations of teamwork and social processes in design, *Design Studies* **16**(2): 143-170.

Cross, N: 2001, Design cognition: results from protocol and other empirical studies of design activity, *in* WC Newstetter (ed), *Design Knowing and Learning: Cognition in Design Education*, Elsevier, Amsterdam.

Cuff, D: 1991, *Architecture: The Story of Practice,* MIT Press, Cambridge, Mass.

Dorst, K: 1995, Analysing design activity: New directions in protocol analysis, *Design Studies* **16**: 139-142.

Eastman, CM: 1968, Explorations of the cognitive processes in design, *in* Department of Computer Science Report, Carnegie Mellon University, Pittsburgh.

Ericsson, KA and Simon, HA: 1993, *Protocol Analysis: Verbal Reports as Data,* MIT Press, Cambridge, Mass.

Fish, JC: 1996, *How Sketches Work: A Cognitive Theory for Improved System Design*, the Loughborough University of Technology.

Gabriel, G and Maher, ML: 1999, Coding and modelling communication in architectural collaborative design, *in* J Bermudez (ed), *ACADIA'99*, pp. 152-166.

Gabriel, G and Maher, ML: 2000, Analysis of design communication with and without computer mediation, *in* A Woodcock (ed) *Collaborative Design: Proceedings of Co-Designing 2000*, Springer-Verlag, London, pp. 329-337.

Gero, JS and Mc Neill, T: 1998, An approach to the analysis of design protocols, *Design Studies* **19**(1): 21-61.

Goldschmidt, G: 1990, Linkography: Assessing design productivity, *in* R Trappl (ed), *Cybernetics and Systems '90*, World Scientific, Singapore, pp. 291-298.

Goldschmidt, G: 1995, The designer as a team of one, *Design Studies* **16**(2): 189-209.

Goldschmidt, G and Weil, M: 1998, Contents and structure in design reasoning, *Design Issues* **14**(3): 16.

Jay, E. S. and Perkins, D. N.: 1997, Problem finding: The search for mechanism, *in* MA Runco (ed), *Creativity Research Handbook*, Hampton, Cresskill, NJ, pp. 257-294.

Kvan, T: 2001, The pedagogy of virtual design studios, *Automation in Construction* **10**(3): 345-353.

Kvan, T, Vera, A and West, R: 1997, Expert and situated actions in collaborative design, i*n* J-P Barthes (ed), *Proceedings of Second International Workshop on CSCW in Design* International Academic Publishers, Beijing, pp. 400-405.

Kvan, T, Yip, WH and Vera, A: 1999, Supporting design studio learning: An investigation into design communication in computer-supported collaboration, *in CSCL'99* Stanford University, pp. 328-332.

Lawson, B: 1994, *Design in Mind,* Butterworth Architecture, Oxford England.

Oxman, R: 1995, Viewpoint Observing the observers: Research issues in analysing design activity, *Design Studies* **16**(2): 275-283.

Reitman, WR: 1964, Heuristic decision procedures, open constraints, and the structure of ill-defined problems, *in* GL Bryan (ed) *Human Judgements and Optimality,* John Wiley, New York.

Robbins, E and Cullinan, E: 1994, *Why Architects Draw,* MIT Press, Cambridge, Mass.

Schon, DA: 1983, *The Reflective Practitioner: How Professionals Think in Action,* Basic Books, New York.

Schon, DA: 1985, *The Design Studio: An Exploration of its Traditions and Potentials,* RIBA Publications, London.

Sellen, AJ and Harper, R: 2001, *The Myth of the Paperless Office,* MIT Press, Cambridge, Mass.

Simon, HA: 1984, The structure of ill-structured problem, *in* N Cross (ed), *Developments in Design Methodology,* John Wiley.

Wigley, M: 2001, Paper, scissors, blur, *in* M Wigley (ed), *The Activist drawing: Retracing Situationist Architectures from Constant's New Babylon to Beyond*Drawing Center, MIT Press, New York, pp. 27-56.

Zeisel, J: 1984, *Inquiry by Design: Tools for Environment-Behavior Research,* Cambridge University Press, Cambridge.

COMPARING ENTROPY MEASURES OF IDEA LINKS IN DESIGN PROTOCOLS

JEFF WT KAN, ZAFER BILDA AND JOHN S GERO
University of Sydney, Australia

Abstract. This paper uses Shannon's entropy of information to measure linkographs of twelve design sessions which involved six architects in two separate design processes. In one they were not allowed to sketch (blindfolded condition), and in the other they were allowed to sketch. A previous study showed that the architects' overall cognitive activity in the blindfolded condition dropped below their activity in the sketching condition, after 20 approximately minutes during the timeline of the design sessions. This study uses entropy measures as a quantitative tool to study this phenomenon. Assuming that moves in a linkograph are the manifestation of ideas and entropy indicates the idea development potential, we tested whether entropy of idea links would also drop after 20 minutes during the blindfolded sessions. The results show that the visuo-spatial working memory load does not have negative effects on idea development.

1. Background

This section presents the necessary background for this study; this includes the construction of linkography, the measurement of linkography using Shannon's entropy, and the theory and method of the blindfolded designing experiment.

1.1. LINKOGRAPH

Linkography was first introduced to protocol analysis by Goldschmidt (1990) to assess design productivity of designers. It is a technique used in protocol analysis to study designers. The design protocol is decomposed into small units called "design moves". Goldschmidt defines a design move as: "a step, an act, an operation, which transforms the design situation relative to the state in which it was prior to that move" (Goldschmidt 1995), or "an act of reasoning that presents a coherent proposition pertaining to an entity that is being designed" (Goldschmidt 1992). A linkograph is then constructed by linking related moves. It can be seen as a graphical network of associated

J.S. Gero (ed.), Design Computing and Cognition '06, 265–284.
© 2006 *Springer. Printed in the Netherlands.*

moves that represents the design session. Figure 1 is a linkograph with 3 moves.

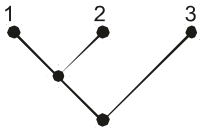


Figure 1. A linkograph with 3 moves, both moves 2 and 3 are related to move 1 but moves 2 and 3 are not related.

The design process can then be looked at in terms of the patterns of moves associations. Goldschmidt identified two types of links: backlinks and forelinks. Backlinks are links of moves that connect to previous moves, in Figure 1 moves 2 and 3 are backlinked to move 1. Forelinks are links of moves that connect to subsequent moves, move 1 is forelinked to moves 2 and 3. Conceptually they are very different: "backlinks record the path that led to a move's generation, while forelinks bear evidence to its contribution to the production of further moves" (Goldschmidt 1995). Link index and critical moves were devised as indicators of design productivity. Link index is the ratio between the number of links and the number of moves, and critical moves are design moves that are rich in links, they can be forelinks, backlinks, or both. In her exposition, design productivity is positively related to the link index and critical moves, that is, a higher value of link index and critical moves indicates a more productive design process.

This link index and critical moves approach is biased towards a highly linked linkograph because a saturated linkograph will have a high value of link index and critical moves. Kan and Gero (2005) argue that a fully saturated linkograph indicates no diversification of ideas, hence less opportunity for quality outcomes.

1.2. INFORMATION THEORY

Following the above argument, empty linked and fully linked linkographs are not interesting, we speculate, intuitively, a randomly linked linkograph embodies a balanced process that embraces integration and diversification of ideas. This directs us to Shannon's (1948) concept of entropy as a measure of information. In Shannon's information theory, the amount of information carried by a message or symbol is based on the probability of its outcome. If there is only one possible outcome, then there is no additional information because the outcome is known. Information is transmitted through recognizable symbols predetermined by the source and the receiver. For example a source is sending out ten ON/OFF signals and one of them is OFF but the others are ON. We can say the probability of a OFF signal, p(OFF),

is 0.1 and the probability of a ON signal, p(ON), is 0.9. Consider the following two cases:

1) If the first signal the receiver gets is OFF (p=0.1) then there is no further transmission needed because the outcome is known. This has the assumptions that the receiver knows the total number of signals (10), the probabilities of the signals (on/off) with the total probability equals 1.

2) If the first signal being transmitted is ON (p=0.9) the receiver is still uncertain of the rest of the signals.

So the transmission of case 1 carries more information and we can see the amount of information carried by a symbol (ON or OFF in this case) is related to the probability of its outcome. Based on this Shannon proposed an information-generating function h(p); in mathematical terms the information function needs to have the following properties:

$h(p)$ is continuous for $0 <= p <= 1$,

$h(p_i) =$ infinity if $p_i = 0$,

$h(p_i) = 0$ if $p_i = 1$,

$h(p_i) > h(p_j)$ if $p_j > p_i$,

$h(p_i) + h(p_j) = h(p_i * p_j)$ if the two states are independent.

Shannnon proved that the only function that satisfies the above five properties is

$$h(p) = - \log(P) \tag{1}$$

Given a set of N independent states a_1, … a_N and corresponding possibilities p_1, … p_N. In our example N=2, p_1=p(ON)=0.9, and p_2=p(OFF)=0.1. He then derived Entropy (H), the average information per symbol in a set of symbols, to be:

$$H = p_1 * h(p_1) + p_2 * h(p_2) + \ldots + p_N * h(p_N)$$

$$\text{Therefore } H = -\sum_{i=1}^{n} p_i \log (p_i) \quad \text{with} \quad \sum_{i=1}^{n} p_i = 1 \tag{2}$$

In our case there are two symbols (on/off) and entropy is expressed by:

$$H = -p(ON)\text{Log}(p(ON)) - p(OFF)\text{Log}(p(OFF)) \tag{3}$$

$$H = -(0.9 * \log_2(0.9) + 0.1 * \log_2 (0.1)) = 0.469$$

The "logarithmic base corresponds to the choice of a unit for measuring information" (Shannon 1948); Here we use base 2 to represent bitwise information per symbol.

Section 2.3 will describe how this can be applied to calculate the entropy of a linkograph.

1.3. BACKGROUND TO SKETCHING AND BLINDFOLDED DESIGNING

Reviewing the literature in design studies and cognitive psychology, we can present two views on imagery and sketching activities in design, which also make a distinction between them. In the first view, sketching directly produces images and ideas externalized on paper and then the designer commences a dialogue with them via their perceptual mechanisms. In this way, the design problem space is explored, and restructured through this dialogue (Schon and Wiggins 1992; Goldschmidt 1992; Suwa et al. 1998). In the second view, during the use of imagery alone for designing (designing without sketching), a designer has to accumulate considerable amount of knowledge/meaning before an image is generated, which suggests concept formation without drawings and without direct perceptual input. As with sketching activity, there is the dialogue with the images to restructure the design space; this probably is constrained within working memory capacities (Logie 1995).

Do architects necessarily start designing with external representations in the early stages of design? Anecdotal examples are often quoted of major architects such as Frank Lloyd Wright who could conceive of and develop a design entirely using imagery with an external representation of the design only being produced at the end of the process (Franklin 2003). This implies it should be possible for some designers to develop and maintain an internal designing activity for a prolonged time. We refer to this activity as the use of imagery alone in designing.

Athavankar (1997) conducted an experiment where an industrial designer was required to design a product in his imagery (with an eye mask on), so that he had no access to sketching and the visual feedback it provides. The study showed that expert designers may be able to use imagery alone in the conceptual design phase, before externalizing their design thoughts. Similar results were obtained in a study with software designers (Petre and Blackwell 1999) where they were required to design using their mental imagery alone. The results of the study are qualitative, informing the possible cognitive processes and mechanisms that might be involved in blindfolded designing.

Bilda et al. (2006) studied architects with an approach similar to Athavankar's (1997) study. At the end of the blindfolded designing, the participants were asked to quickly sketch the design solution they held in

their minds. The solutions were assessed by judges and the results showed that the participants did reasonably well. When the participants were blindfolded, they were able to produce designs by using their cognitive resources to create and hold an internal representation of the design.

In another study, Bilda and Gero (2005) presented the cognitive activity differences of three expert architects when they design in blindfolded and sketching conditions. It was observed that all participants' overall cognitive activity in the blindfolded condition dropped below their activity in the sketching condition approximately after 20 minutes during the timeline of the design sessions. This drop was explained by higher cognitive demands and limitations of visuo-spatial working memory in the blindfolded conditions. The results within the small group of participants showed that sketching off-loaded visuo-spatial working memory.

In this paper we suggest the use of entropy to test whether the working memory limitations have an impact on idea development. Evidence in working memory research supports the suggestion that the cognitive load should be higher in a blindfolded condition since image maintenance and synthesis of images requires more executive control resources (Pearson et al. 1999; Baddeley et al 1998). Other empirical studies on visuo-spatial working memory (VSWM) also show that the capacity of the VSWM is limited when visuo-spatial tasks are done using imagery (Ballard et al. 1995; Walker et al. 1993; Phillips and Christie 1997). Similar results have been obtained for the phonological loop of the working memory, when verbal tasks were performed using imagery (Baddeley 1986). We presume that the design ideas could have a visuo-spatial mode and a verbal conceptual mode in imagery working in parallel. Since there is a working memory limitation in using imagery, the idea development could slow down after a while during the timeline of the blindfolded designing activity. On the other hand sketching activity could support and improve the idea development activity, since working memory is off-loaded continuously.

2. Method

The six architects who participated in the study (2 female and 4 male) have each been practicing for more than 15 years. Architects A1 and A2 run their own companies and have been awarded prizes for their designs in Australia; Architect A3 is a senior designer in a well-known architectural firm. The three participants were teaching part-time in design studios. A4 works for one the Australia's largest architectural companies and he has been the leader of many residential building projects from small to large scale. A5 is one of the founders and director of an award wining architectural company. A6 is a very famous residential architect in Sydney, and he directs his company known by his name with 50 employees.

2.1. DESIGN OF THE EXPERIMENTS

We conducted think-aloud designing experiments with the six architects. The first group of the three architects, A1, A2 and A3 were initially engaged in a design process where they are not allowed to sketch. This phase is called the experiment condition where they received design brief 01. One month after the experiment condition the same three architects were engaged in a design process where they are allowed to sketch. This phase is called the control condition where they receive design brief 02. Design brief 01 requires designing a house for two artists: a painter and a dancer. The house is to have two studios, an observatory, a sculpture garden and living, eating, sleeping areas. Design brief 02 requires designing a house on the same site as design brief 01 but this time for a couple with 5 children aged from 3 to 17, a house that would accommodate children and parent sleeping areas, family space, study, guest house, eating and outdoor playing spaces.

The second group of the three architects was first engaged in the sketching (control condition) session, where they received design brief 02. Then after one month they were under the experiment condition working on design brief 01, where they were not allowed to sketch.

The set-up of the study for both experiment and control conditions has a digital video recorder with a lapel microphone, directed at the designer. In the experiment condition, the architects were required to put on a blindfold and start thinking aloud, Figure 2(a). At the end of the 45 minute session they were asked to take off the blindfold and quickly sketch what they held in their minds in the allowed 5 minute period. They were instructed that changes were not permitted to the design solution they originally had in their minds. The details of the experimental procedure can be found in Bilda et al (2006). During the 45 minute period of the sketching condition, the architects were required to think aloud and sketch at the same time.

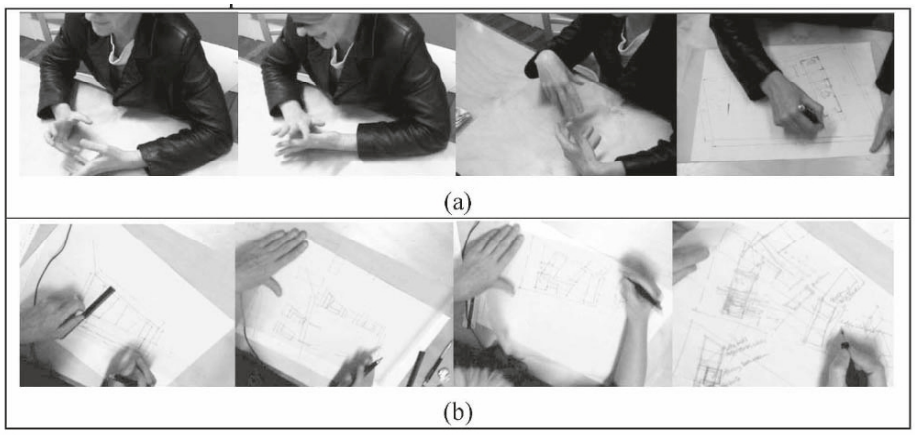


Figure 2. (a) Blindfolded session followed by quick sketching, (b) sketching session.

2.2. CODING THE LINKS

The linkography technique involves dividing the protocol into design moves (Goldschmidt 1992) and looking at the design process in terms of relationships created by the links between those moves. In this study we segmented the protocols according to designers' intentions (Suwa, et al. 1998). Suwa et al. (1998) discussed that this segmentation technique was similar to the notion of design moves, thus we used the same segmented intervals in the process of coding links.

The process of connecting the ideas between the segments is carried out in two runs. In the first run, the coder starts from the first segment and sequentially connects the ideas, and revisited meanings between the segments. The analyzer relies on the verbalization only while linking the ideas in the blindfolded designing protocols.

During linking the ideas of sketching protocols, video footage for each segment was visited as well. Linking the ideas in protocols could be difficult task, if the analyzer loses track of the ideas developed previously along the timeline of the protocol session. In order to prevent missing links and for linking the ideas more reliably, we employed a technique that involves a word search in order to detect the words used more frequently such that the analyzer ends up with a list of frequently repeated words in each segment. The next stage is browsing through the selected segments to confirm that the words are used in the appropriate context. If the ideas are related, then these segments are connected to each other. This procedure helps us to connect the meanings, which are distant from each other and which might have been missed in a sequential analysis.

2.3. ENTROPY MEASUREMENT

In our measure of linkograph entropy we are interested not only in the number of linked moves but also the distribution of those links. Two extremes example are: an empty linked (none of the moves is related) and a fully linked (all the moves are linked) linkograph. An empty linked linkograph can be considered as a non-converging process with no coherent ideas and a fully linked linkograph stands for a total integrated process with no diversification. In both cases the opportunities for idea development are very low. This line of reasoning can be expressed in terms of entropy; if we randomly pick a move in an empty linked linkograph we can be sure that it has no links. This sounds obvious but if we consider this linkograph as a carrier with zero information content, because the outcome is known, it will have zero entropy. Similarly, a fully linked linkograph will also have zero entropy.

In order for entropy measurement of linkograph to be meaningful, we follow Kan and Gero's (2006) measurement method, which is based on the

conceptual difference of forelink, backlink, and horizontal link (called "horizonlink"). Entropy is measured in rows of forelinks, backlinks, and horizonlinks separately. Consider an abstracted linkograph with four moves as in Figure 3. The linkages between moves are denoted by back dots and the grey dots denoted possible linkages but not linked. We calculate forelink entropy for each move except the last two moves. In Figure 3(a) move 4 will not have forelinks and move 3 could either be linked or unlinked to move 4 which will have zero entropy. Similarly each move, except the first two, will receive a backlink entropy, Figure 3(b). Moves can be linked to a previous move (n-1) or moves further apart, Figure 3(c).

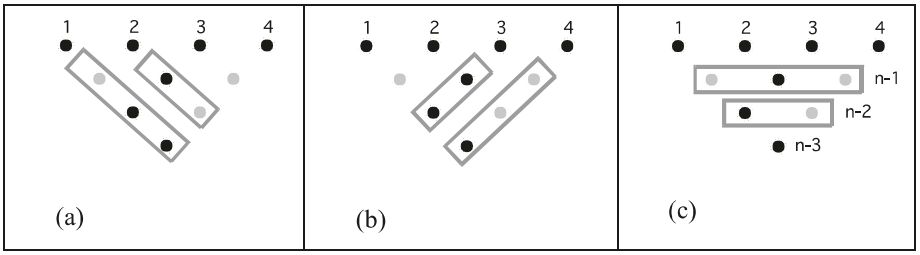


Figure 3. Abstracted linkograph for entropy measurement, back dots denote linkage between moves. (a) Measuring entropy of forelinks of each row, (b) measuring entropy of backlinks of each row, and (c) measuring entropy of horizonlinks.

Moves residing in working memory will usually have high interconnections, this we refer as the cohesiveness of moves. Links that connect moves that are far apart, those that are not in working memory are considered as incubated moves.

Forelink entropy in Figure 3(a) is contributed to by moves 1 and 2 which are marked out with rectangles. In move 1 there are three total nodes for links; moves 1 and 2 are unlinked while move 1 is linked to moves 3 and 4. The percentage for linked nodes is 66.6% and the percentage of unlinked node is 33.3%. If we consider linked nodes as ON and unlinked nodes as OFF, the probability will be p(ON)=0.666 and p(OFF)= 0.333 respectively.

This case is same as the example in Section 1.2 so we can use formula (3), forelink H for move 1 becomes:

$$- 0.666\text{Log}(0.666)-0.333\text{Log}(0.333) =0.918$$

So for move 2 the forelink H is:

$$-0.5\text{Log}(0.5) - 0.5\text{Log}(0.5)=1$$

As we see for move 3 there is only one possible link so whether it is ON or OFF the entropy is zero because Log(1) = 0.

Using this method, in Figure 3(b), the backlink entropies for moves 3 and 4 are zero and 0.918 respectively.

For the horizonlink entropy we consider, in this case, the two rows: n-1 and n-2. Using formula (3) we can get the entropy of n-1 row to be 0.918 and the entropy for n-2 row is 1. Since people have limited short term memory (Miller 1956), if we follow Miller's "magic number seven plus or minus two" objects the linkographs seldom have links with more than 9 moves and the links with far apart moves will decease unless that move is an important move. Figure 4 shows a typical decrease of links, which has a lot of cohesive links and little incubated links. As both fully cohesive links and no incubation link will score zero in horizonlink entropy, our measure of horizonlink entropy will basis toward having incubated move links and not saturated cohesiveness links.

Kan and Gero (2006) concluded that forelink entropy measures the idea generation opportunities in terms of new creations or initiations. Backlink entropy measures the opportunities according to enhancements or responses. Horizonlink entropy measures the opportunities relating to cohesiveness and incubation.

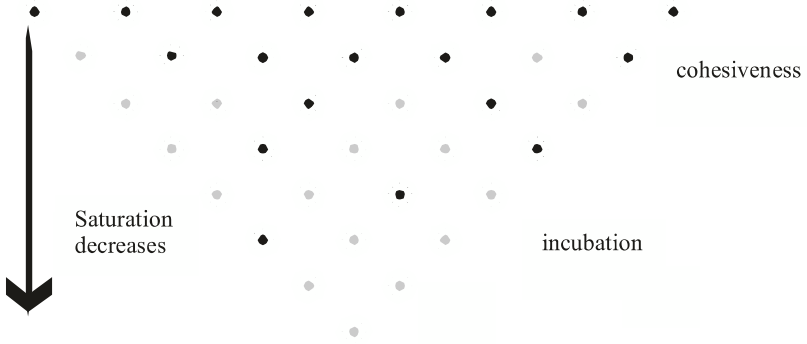


Figure 4. A linkography with typical distribution of links in during a design process.

3. Results

A previous study, using the results from the first group of architects, showed that the overall cognitive activity in the blindfolded condition dropped below the activity in the sketching condition approximately after 20 minutes into the design sessions (Bilda and Gero 2005). This drop in performance can be explained by higher cognitive demands and limitations of VSWM in blindfolded conditions. However, with this drop of cognitive activity, in the blindfolded condition the architects can still produce satisfactory designs. Their design outcomes, sketches, were judged by a qualified jury of three designers. Surprisingly, all the blindfolded sessions received a higher score

as compared to the corresponding sketch sessions. We attempt to explore these phenomena by using entropy measures of the linkograph.

3.1. LINKOGRAPHS OF THE SESSIONS

After coding the links, 12 linkographs of the sessions had been produced; they exhibit different patterns that reflect different design processes. For example the linkograph in Figure 5(a) reflects a relatively holistic design process (the linkograph is well integrated) while the linkograph in Figure 5(b) represents a process of trying out different options (there are obvious clusters in the linkograph). This observation was confirmed with our qualitative analysis.

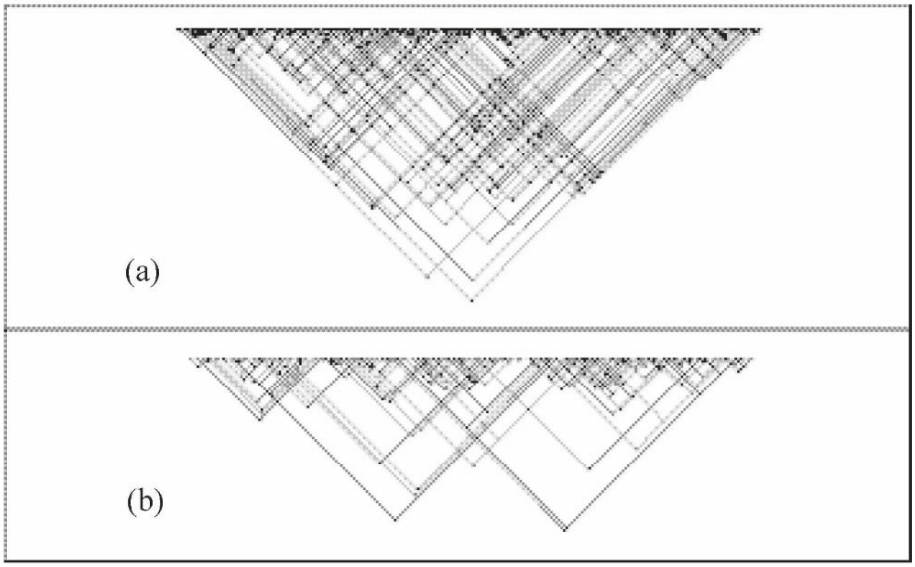


Figure 5. (a) Linkograph of a sketch session, (b) linkograph of a blindfolded session.

The detailed qualitative analysis of the 12 linkographs is beyond the scope of this paper. Here we examine whether the drop of cognitive activity in the blindfolded session is reflected in the entropy of the linkograph.

Looking at the linkograph, some of the blindfolded sessions seem to suggest a more productive process according to Goldschmidt's definition due to the visual density of links. However, we cannot track down where the cognitive activity started to drop.

Then we count the moves of the first 20 minutes in all the sessions as compared to the rest of the session, Table 1. In general the blindfolded sessions do have slightly more moves in the first half of the session, this is consistence with the previous result as cognitive activity is reflected in moves, but visually the number of links does not seem to drop in the second

half. The average and standard deviation in Table 1 are for reference only, each case should be looked at individually.

TABLE 1. Number of moves in the first 20 minutes and the rest of the session.

	Blindfolded No. of Moves			**Sketch No. of Moves**		
	20 min	Rest	Total	20 min	Rest	Total
Architect 1	89	78	167	68	77	145
Architect 2	63	91	154	77	107	184
Architect 3	87	82	169	65	77	142
Architect 4	92	75	167	74	95	169
Architect 5	73	72	145	91	62	153
Architect 6	69	53	122	71	101	172
Average	***78.83***	***75.17***	***154.00***	***74.33***	***86.50***	***160.83***
Standard Deviation	*12.04*	*12.70*	*18.26*	*9.20*	*17.22*	*16.70*

3.2. ENTROPY OF THE SESSIONS

We use formula (3) to calculate the forelink and backlink entropy of each move in a linkograph. In order to get a picture of the whole session, we add the entropy of each move to get the total forelink and backlink entropy of a session. Likewise, we get the total horizonlink entropy by adding the entropy of all the horizontal rows. We divide the entropy by the total number of moves in that session to normalize them. Table 2 shows the normalized results of entropy of each session together with their link index. Overall the blindfolded sessions have higher entropy and link index in both groups. Investigating individual architects, only the first architect in the first group does not follow this trend.

Table 2 shows that the first group of three architects' blindfolded (BF) and sketch (SK) conditions have very similar link indices (1.39, 1.36) and entropy values (0.343, 0.336). For the second group of architects, the link index and total entropy values are higher in the BF condition compared to their SK condition (0.511, 0.441 and 2.39, 1.95) with a relatively small standard deviation in the BF condition. These values are not generalizable but are used for exploration. This difference could be due to the experimental condition where the second group received the sketching exercise first and the blindfolded exercise later. The second group had an increased familiarity with the problem space, including the site geometry and the environmental factors around the building. This familiarity with the problem space could have improved the second group's potential for idea development, hence higher entropy.

In general the backlink entropy is higher than the forelink entropy, with the exception of one sketch condition and one blindfold condition. Forelink

entropy indicates idea generation potential based on new initiation which may reflect creativity. The architect who had the forelink entropy higher than the backlink entropy in the sketch session received the highest rank by the judges in the categories of "Innovative" and "Creative". The relationship between forelink entropy and creativity needs further investigation and is inconclusive at this moment.

It is easier to use fixed number of moves as reference as the procedure can be automated. Figure 6 shows A1's backlink, forelink and horizontal link entropy changes, using 28 moves as a window width. Figure 7 shows the corresponding linkographs.

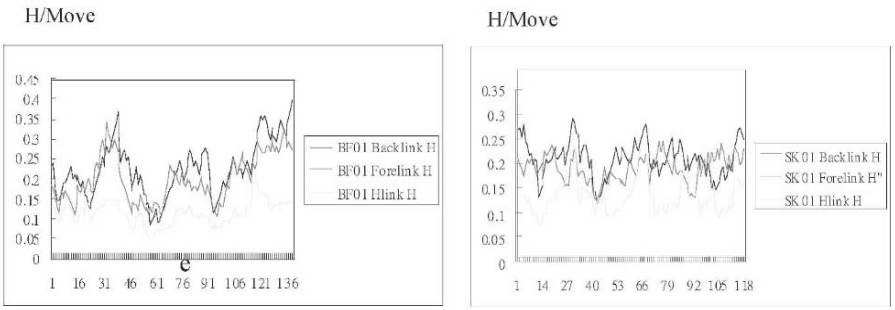
TABLE 2. Entropy and link index of each session.

		BH/ moves	**FH/ moves**	**HH/ moves**	**Total**	**Link Index**
Blindfolded (BF)	BF 1	0.125	0.122	0.060	0.307	1.20
	BF 2	0.161	0.155	0.066	0.383	1.68
	BF 3	0.143	0.140	0.055	0.338	1.28
	av	***0.143***	***0.139***	***0.060***	***0.343***	***1.39***
	sd	*0.018*	*0.017*	*0.006*	*0.038*	*0.257*
	BF 4	0.240	0.220	0.093	0.553	2.48
	BF 5	0.224	0.193	0.082	0.499	2.18
	BF 6	0.188	0.189	0.105	0.481	2.50
	av	***0.217***	***0.201***	***0.093***	***0.511***	***2.39***
	sd	*0.027*	*0.017*	*0.012*	*0.037*	*0.179*
Sketching (SK)	SK 1	0.137	0.124	0.077	0.337	1.41
	SK 2	0.157	0.150	0.065	0.373	1.48
	SK 3	0.124	0.131	0.044	0.299	1.20
	av	***0.139***	***0.135***	***0.062***	***0.336***	***1.36***
	sd	*0.017*	*0.013*	*0.017*	*0.037*	*0.146*
	SK 4	0.227	0.203	0.098	0.529	2.41
	SK 5	0.176	0.125	0.071	0.372	1.68
	SK 6	0.184	0.175	0.063	0.422	1.76
	av	***0.196***	***0.168***	***0.077***	***0.441***	***1.95***
	sd	*0.027*	*0.040*	*0.018*	*0.080*	*0.400*

BH: entropy of back links FH: entropy of forelinks HH: entropy of horizonlinks

We can identify two regularities in these graphs: the relationship between the three types of link entropies and the overall trend. When we look at the relationships between the backlinks, forelinks and horizontal links, we can

observe similar relationships but with different shapes for all architects and conditions. In order to study the trends, we use polynomial fit as a tool to explore.



(a) (b)

Figure 6. Change of entropy (a) in BF condition (b) in SK condition.

Figure 8 shows the overall trends for all the twelve sessions with quadratic (second degree) polynomial fit of the total normalized entropy. A quadratic curve fit is to find the closest quadratic equation to represent the trend of the data. There are two basic shapes of a quadratic equation, a "n" or an "u" shape. The "u" shape trend suggests that the entropy values are climbing toward the end of the session, whereas the "n" shape trend suggest the opposite. It is observed that most of the architects in this experiment have the same trend in the BF and SK session. Can this represent the style of thinking?

3.3. HYPOTHESIS TESTING

In this section we test if the idea development drops 20 minutes into the blindfolded design process, due to working memory limitations. In the previous sections, in Table 1, we can see the total number of moves dropped after 20 minutes in the BF session but looking at the entropy trends, in Figure 8, some of them rise towards the end of a session.

If we assume idea development is positively correlated to entropy and use the move at 20 minutes to divide the linkograph, we can test if the idea development potential drops as the cognitive activities drops. Figure 9 shows how we calculate the entropy of the first and second half of the linkographs. We check if there is a significant differences between the BF and SK conditions. We chose 20 minutes as a demarcation, based on a previous study, where the first three participants' cognitive performance in BF conditions dropped below their performance in SK conditions, after around 20 minutes (Bilda and Gero 2005).

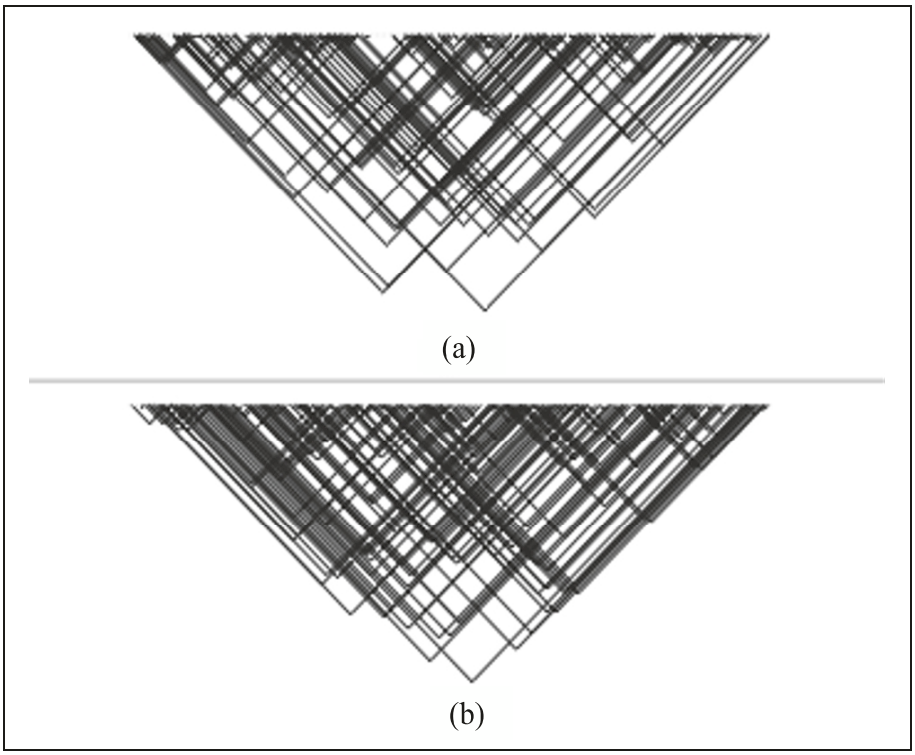
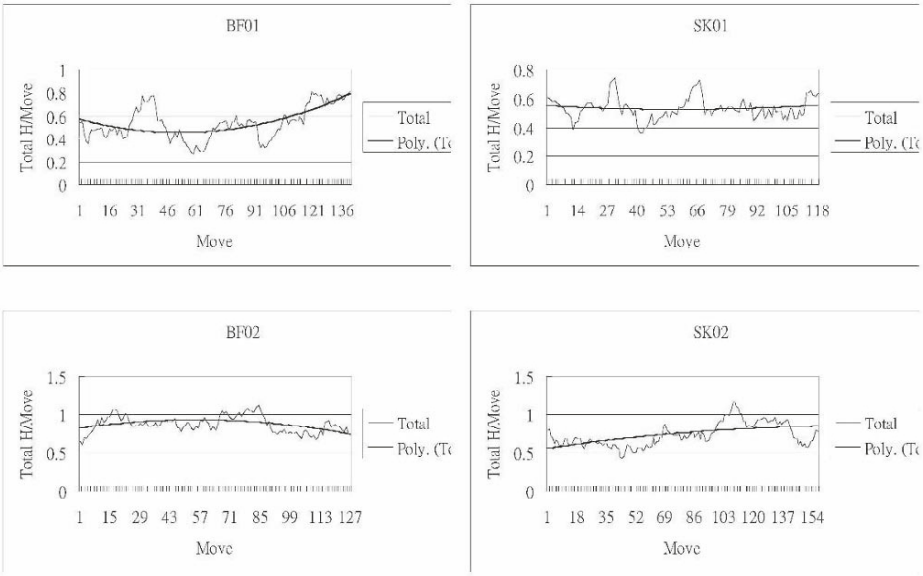


Figure 7. (a) Linkograph of the BF session, (b) linkograph of the SK session.



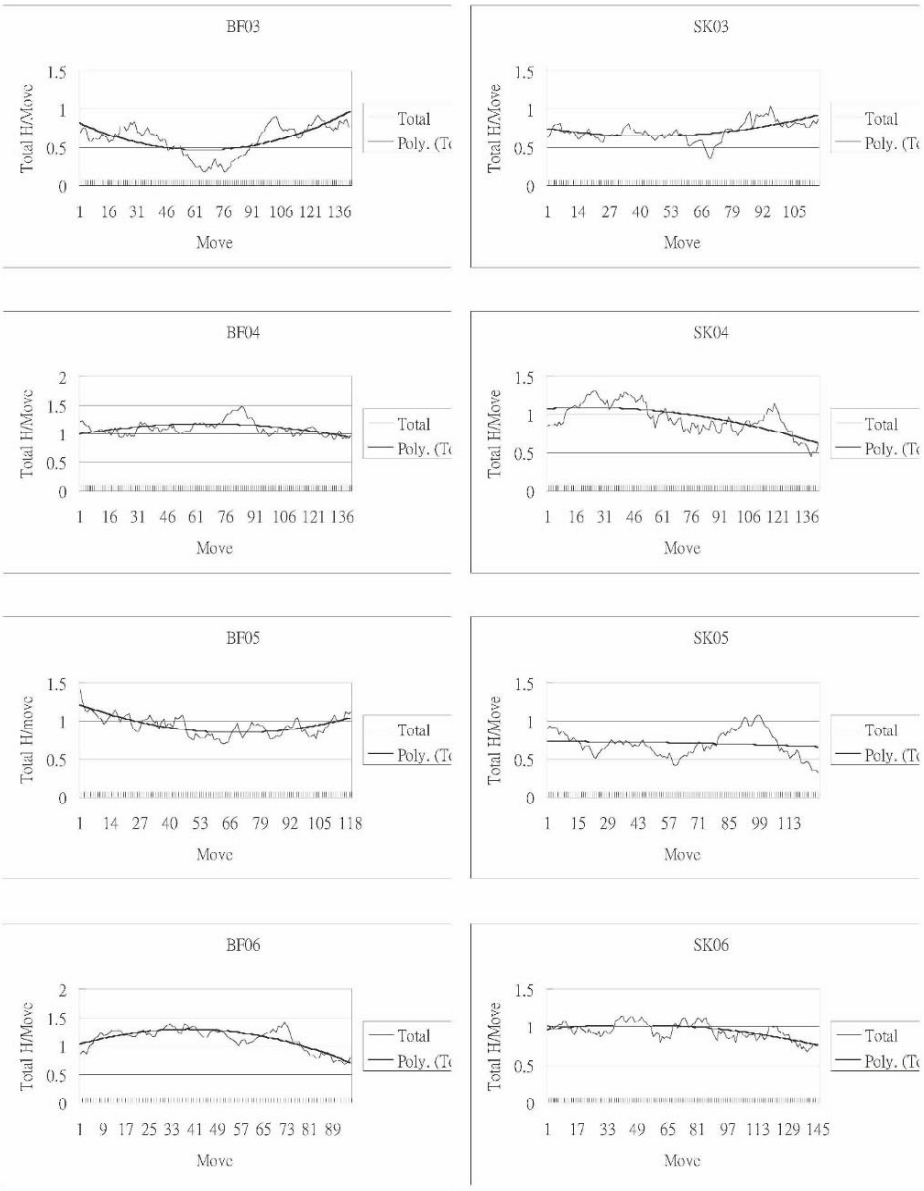


Figure 8. Polynomial fit of the total entropy for all the sessions.

In this calculation we ignore those linked nodes outside the shaded triangles in Figure 9. The results in Table 3 show that in both conditions there is a drop in entropy in the second half of the sessions. In the sketch

condition five out of the six has lower entropy in the second half; conversely half of blindfolded sessions have higher entropy in the second half.

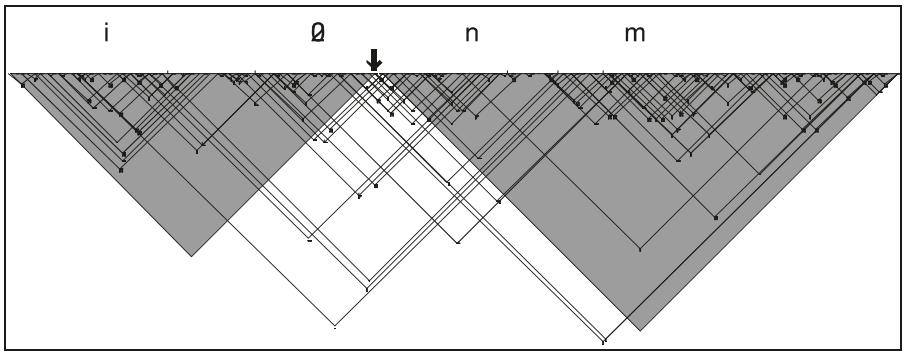


Figure 9. Calculating the entropies of the first and second half of the linkograph; only links within the gray triangle are counted.

Generally, the entropy per move of the second half of the session is lower than the first half. This parallels our assumption that entropy measures the opportunities for idea generation. In the second half, designers are approaching the end of a session and the design converges to a particular approach, therefore fewer opportunities occur for idea development. Consequently the entropy values would be lower in the second half compared to the first. However, what account for the increases of entropy in three of the BF sessions?

TABLE 3. Total normalized entropies of each session.

	Blindfolded Entropies/Move		**Sketch Entropies/Move**	
	First 20 min	The rest	First 20 min	The rest
Architect 1	0.343	0.411	0.450	0.398
Architect 2	0.650	0.498	0.484	0.448
Architect 3	0.406	0.441	0.461	0.518
Architect 4	0.743	0.670	0.783	0.588
Architect 5	0.911	0.679	0.603	0.529
Architect 6	0.857	0.875	0.652	0.540

In order to understand what is happening to the idea links refer to formula (3) which Kan and Gero (2006) plotted as shown in Figure 10.

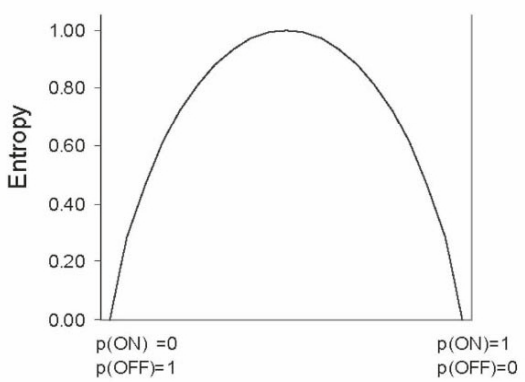


Figure 10. Maximum entropy when p(ON)=p(OFF)=0.5.

Figure 10 shows that when *p*(ON) is between {0.35, 0.65}, entropy H is over 0.93 that is if the links in a row are between 35% and 65% saturated it will produce a very positive value. If the links are less then 5% or over 95%, it will produce a very low H value (below 0.29). Either heavily linked or rarely linked second half linkographs will have low entropy values. Reviewing the 12 linkographs, we could conclude they all fit the second case; that is linked ideas are sparse. So a higher value of entropy implies more linkage among moves, hence better development of ideas. Accordingly three of the BF sessions have better inter-connectivity in the second half of the session. This result suggests that in the blindfolded condition, working with limited VSWM, does not have a serious impact on the inter-connectivity of ideas as compare to the sketch condition. This is confirmed by two tailed paired t-test with the null hypothesis being: there is a drop of entropy after 20 minutes in the blindfolded sessions. To test against the hypothesis we compute the entropy difference of the first 20 minutes and the rest of the session, Table 4. Then we paired the two sets of data according to the architect.

TABLE 4. Entropy differences.

	Pair 1	Pair 2	Pair 3	Pair 4	Pair 5	Pair 6
BF diff	-0.068	0.152	-0.036	0.072	0.232	-0.017
SK diff	0.053	0.036	-0.057	0.195	0.074	0.112
BF: Blindfolded condition SK: Sketch condition diff: entropy of the first 20 min – entropy of the rest						

The t-test resulted is 0.82, so the hypothesis is not supported for p =< 0.05. The development of linked idea does not drop half way through the blindfolded design process. The implication of this result is profound,

contrary to usual beliefs; when VSWM is overloaded it may not have negative effects on idea development.

4. Discussions and Conclusions

Kan and Gero (2005) showed that entropy could be a useful tool to measure idea development, because it measures the chances of linking ideas to previous ones (backlinks) and the chances of developing ideas that would be sought later (forelinks) during the design process. We interpreted this linkograph entropy measure as "the potential for idea development". Then we tested whether working memory limitations has an impact on the potential for idea development.

The results showed that the idea development did not necessarily slow down after some time during the use of imagery alone. In our previous findings, perceptual activity of the three architects significantly slowed down after 20 minutes during the BF designing sessions. It was improved during sketching (Bilda and Gero 2005). Perceptual activity, being related to the tasks in VSWM, slowed down because of VSWM limitations. Perhaps, idea development is not dependent on a visuo-spatial modality, since idea development potential seemed not to be affected by working memory load.

Bilda and Gero (2005) also observed that the variances in functional activity (attaching meaning and purpose to things) were similar in BF and SK conditions. This implied that sketching did not add to (improve) production of meaning, however it improved perceptual activity. This result also suggested that the cognitive load (working memory load) might be related to perceptual activity rather than functional activity. The visuo-spatial tasks which require executive resources in working memory might create the cognitive load, but not necessarily the concept/meaning formation. Assuming that idea development in imagery is concept-related rather than figure-related, results in this paper support our previous findings. We conclude that that the VSWM load does not have negative effects on idea development. Idea development could be dependent more on a conceptual modality than the visuo-spatial modality.

Entropy measures provide a formal basis for the evaluation of linkographs derived from protocols of designers. Whilst they are still in their infancy we have shown how measuring the entropy of different views of linkographs has the potential to provide the basis for what would otherwise be a weak qualitative assessment.

Acknowledgements

This research supported by an International Postgraduate Research Scholarship and a University of Sydney International Postgraduate Award, facilities are provided by the Key

Centre of Design Computing and Cognition. We would like to thank to the architects who participated in this study.

References

Akaike, H: 1973, Information theory as an extension of the maximum likelihood principle, *in* B Petrov and C Csaki, Akademiai Kiado (eds), *Second International Symposium on Information Theory*, Budapest, pp. 267-281.

Athavankar, UA: 1997, Mental imagery as a design tool, *Cybernetics and Systems* **28**: 25-47.

Baddeley, AD: 1986, *Working Memory*, Oxford, Clarendon Press.

Baddeley, AD, Emsile, H, Kolodny, J and Duncan, J: 1998, Random generation and executive control of working memory, *Quarterly Journal of Experimental Psychology* **51A**: 819-852.

Ballard, DH, Hayhoe, MM, and Pelz, JB: 1995, Memory representations in natural tasks, *Journal of Cognitive Neuroscience* **7**: 66-80.

Bilda, Z , Gero, JS and Purcell AT: 2006, To sketch or not to sketch: That is the question, *Design Studies* (to appear).

Bilda, Z and Gero, JS: 2005, Does sketching off-load visuo-spatial working memory? *in* Gero, JS and Bonnardel, N (eds), *Studying Designers'05*, Key Centre of Design Computing and Cognition, University of Sydney, pp. 145-159.

Franklin, T: 2003, *Falling Water Rising: Frank Lloyd Wright, E.J. Kaufmann, and America's Most Extraordinary House*, New York, Knopf.

Goldschmidt, G: 1990, Linkography: assessing design productivity, *Cyberbetics and System '90*, R Trappl. World Scientific, Singapore, pp. 291-298.

Goldschmidt, G: 1992, Criteria for design evaluation: a process-oriented paradigm, *in* YE Kalay (ed), *Evaluating and Predicting Design Performance*, New York, John Wiley & Son, Inc. pp. 67-79.

Goldschmidt, G: 1995, The designer as a team of one, *Design Issue* **16**(2): 189-209.

Kan, WT and Gero, JS: 2005, Can entropy indicate the richness of idea generation in team designing? *in* A Bhatt (ed), *CAADRIA05* Vol 1, TVB, New Delhi, India, pp. 451-457.

Kan, JW and Gero, JS: 2006, Acquiring information from linkography in protocol studies of designing, *Design Studies* (submitted).

Logie, RH: 1995, *Visuo-Spatial Working Memory*, Hillsdale, Lawrence Erlbaum Associates, Publishers.

Miller, GA: 1956, The magical number seven, plus or minus two: Some limits on our capacity for processing information, *Psychology Review* **63**: 81-97.

Pearson, DG Logie, RH and Gilhooly, KJ: 1999, Verbal representations and spatial manipulations during mental synthesis, *European Journal of Cognitive Psychology* **11** (3): 295-314.

Petre, M and AF Blackwell: 1999, Mental imagery in program design and visual programming, *International Journal of Human-Computer Studies* **51**(1): 7-30.

Phillips, WA and Christie, DFM: 1997, Components of visual memory, *Quarterly Journal of Experimental Psychology* **29**: 117-133.

Schon, D A and Wiggins G: 1992, Kinds of seeing and their functions in designing, *Design Studies* **13**(2): 135-156.

Shannon, CE: 1948, A mathematical theory of communication, *The Bell System Technical Journal* **27**: 397-423.

Suwa, M, Purcell, T and Gero, JS: 1998, Macroscopic analysis of design processes based on a scheme for coding designers' cognitive actions, *Design Studies* **19** (4): 455-483.

Walker, P, Hitch, G, and Duroe A: 1993, The effect of visual similarity on short-term memory for spatial location: Implications for the capacity of visual short term memory, *Acta Psychologica* **83**: 203-224.

ANALYSING THE EMOTIVE EFFECTIVENESS OF RENDERING STYLES

RAJI TENNETI AND ALEX DUFFY
University of Strathclyde, UK

Abstract. Computer graphics images are characterised by both object information and emotive implications. To promote proper interpretation, it is important to convey incomplete or approximate object information in conceptual design, as well as emotive expressiveness, via the graphics interface. This paper presents a study of user perception and emotional responses to different rendering styles using Kansei Engineering. The investigation involved a sample comprising of 61 students and faculty, and 30 different rendering styles representing existing photorealistic (PR), non-photorealistic (NPR) and new vague rendering (VR) styles. The study has shown that VR styles are able to affect viewers of images in a different way than PR and NPR styles. That is, VR styles are most effective for conveying *affective* and *functional* content, PR styles for *affective* content, and NPR styles for *affective*, *motivational* and *cognitive* content.

1. Introduction

During conceptual design, a designer's geometric design can gradually evolve from its original vague geometric concept, in the form of sketches, towards a more detailed design and eventually to the specification of the final artefact (Stevenson et al. 1996). At the end of this process, these sketches are often transferred into or directly developed in existing Computer Aided Design (CAD) systems to define a model for refining, detailing, analysing, and passing to downstream processes. Computational conceptual design can be considered as a loop between the following activities, Figure 1:

1. *Communication* of the designer with the computer: The designer's mental model is used to initialise a computer model in a CAD system. Mental models are the internal representations of the designer's ideas, concepts or intentions (Burns 2001). These representations involve functional as well as emotional aspects of the final artefact.

J.S. Gero (ed.), Design Computing and Cognition '06, 285–304.
© 2006 *Springer. Printed in the Netherlands.*

2. *Generation* of computer model: The initial stage of this process is usually modelling and consists of production of an abstract description of the concept in 2D or 3D form.
3. *Rendering* uses the description produced by the modeller for presenting visual information. For three-dimensional models the rendering process typically consists of lighting calculations, hidden surface removal, and scan conversion.
4. *Visualisation* and *interpretation* of the displayed image: The displayed image is visualised and interpreted by designers. This involves interactions of cognitive processes in the human brain to form an understanding of the visual information displayed. These interactions of cognitive processes can be explained in terms of a cognitive model such as Interacting Cognitive System (ICS) (Barnard and May 1993). On interpreting the displayed image, the designers form a view of the displayed image in their mind (the mental model of the visualised computer model).
5. *Comparison*: This view is then compared with the original mental model the designers already have in their mind to judge how accurately the images reflecting their intentions are in accordance with the mental model. Based on the view, if their intentions are not accurately modelled in the image generation process, they make changes to the computer model.
6. *Iteration*: During this process the designer's original mental model changes due to an evolving understanding in the process of creation/modification and evaluation of concepts (Smithers 1998). Steps 1 to 5 are followed, until the designer is satisfied with the resulting generated image.

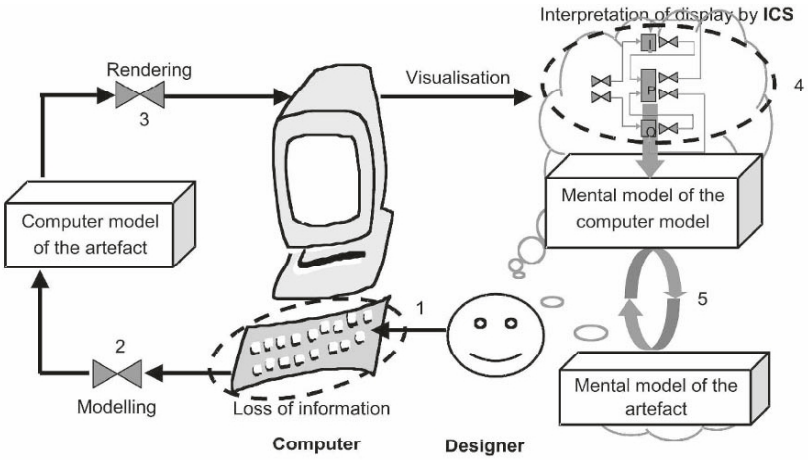


Figure 1. Human-Computer Symbiosis in conceptual design (adapted from Kuczogi et al. 2000).

During conceptual design, the object (representation that concerns shape, size, location, and orientation in an artefact) information is often approximate or incomplete (Guan et al. 1996). Whilst the process of generating images in computer graphics requires specific and precise information. Thus, during the design modelling loop, depicted in Figure 1, the original intent of the designer and the inherent vague information in the design concepts, are often lost.

Central to computer graphics imaging is *object information* and *functional realism* (or fidelity of the information the image provides, Ferwada 2003). *Soft functionality* (McDonagh et al. 2002), a specific form of functionality, includes emotional and intangible aspects such as judgement, inference and articulation. Such soft functionalities are difficult to express objectively. In practise, the generated image has to provide a balance between functional realism, object information, and emotional expressiveness.

Problems in design arise when the images reflecting the designer's intentions are not in accordance with their mental model, or when the designer's intentions are not accurately modelled in the model generation process, resulting in inappropriate images. The focus of the research presented in this paper was on *rendering* (Step 3), and *visualisation and interpretation* of the rendered image (Step 4), for symbiotic interaction between human designers and computers by:

- Conveying approximate or incomplete information.
- Supporting both functional and soft functional values. That is incorporating aspects that conveys emotional as well as object information.
- Minimising the loss of information during the design loop.

Specifically, the research focus was to determine what are the most effective rendering styles for emotive expressiveness of an image. This required an understanding of emotive implications and a means to evaluate the effectiveness of rendering styles. Interacting Cognitive Systems (ICS) (Barnard and May 1993) was used to describe how visual information in an image is processed and understood by humans. The effectiveness of rendering styles was evaluated by using Kansei Engineering (Nagamachi 1995). This involves:

- Identifying the emotive implications of images.
- Evaluation of different rendering styles by mapping the identified emotive implications.
- Statistical analysis of the evaluation data.

To identify the emotive implications of images and evaluate different rendering styles a study was conducted in the form of a questionnaire, the results of which are discussed in this paper. Section 2 of the paper details related work of rendering, conceptual understanding of rendering from cognitive perspective using ICS, and Kansei Engineering. Section 3 presents the experimental methodology for identifying emotive implications of

images and evaluating different rendering styles, and Section 4 presents the results of statistical analysis. Section 5 discusses the work and finally Section 6 concludes with the key findings. The study identifies some specific rendering styles to stimulate emotive implications of images.

2. Related Work

There are a plethora of rendering techniques that are popular in the computer graphics community (Schofield 1994). Photorealistic and non-photorealistic are the two main rendering techniques. Photorealistic techniques aim to create realistic images that mimic physical reality and often aim to be indistinguishable from photographs of real world scenes or objects. Despite much progress of established photorealistic techniques in efficiency and quality, such as ray trace (Kajiya 1986) and radiosity (Heckbert 1990; Sillion 1994), the resulting images may not always be the best way to present information such as ill-defined concepts often found in design. Non-photorealistic rendering is now emerging as an alternative. Non-photorealistic techniques create images and illustrations that do not mimic physical reality but the style and quality of human artist renditions. These techniques use a diverse range of styles from painting (Meier 1996; Curtis et al. 1997; Litwinowicz 1997; Salisbury et al. 1997), pen-and-ink (Hsu et al. 1993; Winkenbach and Salesin 1994), technical illustrations (Dooley and Cohen 1990; Saito and Takahashi 1990; Gooch et al. 1998) through to cartoon type shading (Dooley and Cohen 1990).

Realistic rendering requires a precise and definite geometric model. Such rendering conveys this precision and a sense of exactness. Therefore, this technique has hitherto been considered to be most effective towards the end of design but not to investigate and inspire different ideas or concepts, despite the obvious advantages of having rendered representations at the earliest opportunity. The challenge, therefore, is to discover how to present and maintain the inherent vagueness of ill-defined design concepts. Vague rendering is the rendering of vagueness or the graphical presentation of vagueness in an artefact (Tenneti and Duffy 2005), where vagueness is defined as geometric information that is incomplete and approximate (Guan et al. 1996). Vague information using different modelling methods have varied from the traditional fuzzy set (Yamaguchi et al. 1992), to constraint and parametric-based (Guan et al. 1997), exact modelling (Erwig 1997), particle-systems (Rusak et al. 2000; Horvath et al. 2000), through to probability theory based modelling (Lim 2002). While the generation of vague rendering using vague modelling techniques is largely a technical matter, their acceptance as useful forms of image generation is investigated in this research.

A series of experiments has been conducted (Schumann et al. 1996; Duke et al. 2003) to explore the effect of rendering styles, especially non-photorealistic, on interpretation. Nevertheless there are no studies evaluating

vague rendering styles and also hardly any other studies evaluating photorealistic and non-photorealistic rendering styles.

Purcell and Gero (1998) noted that researchers in design have started to relate their work to a number of areas of research in cognitive psychology and cognitive science. They reviewed the role of sketching to three of these areas such as working memory, imagery interpretation, and mental synthesis. The works of Duke et al. (2003) made a re-appraisal of the relationship between cognitive psychology and computer graphics. Similarly Halper et al. (2003) proposed the necessity of developing a theory of psychology with non-photorealistic rendering. A cognitive model, Interacting Cognitive Systems (ICS) has been used in the context of rendering to contribute to the understanding and interpretation of images (Duke et al. 1999).

2.1. INTERACTING COGNITIVE SYSTEM (ICS)

As can be seen in Figure 1, a rendering system acts as a mediator between the computer world (data of the computer model) and the perception of the model (the output as seen by the designer) (Schofield 1994). The visualisation and interpretation of the displayed information by rendering involves interactions of cognitive processes. Cognition in ICS is viewed as an activity distributed between nine subsystems, each specialised for handling a different type of information. Four of these subsystems mediate the visual perception of images, Figure 2.

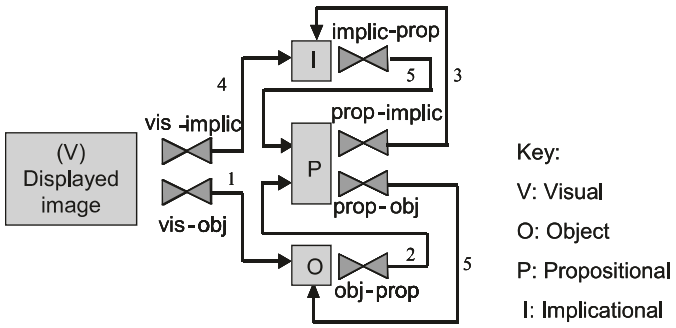


Figure 2. ICS resources for perception (adapted from Herman and Duke 2001).

The interactions between these four subsystems contribute to form an understanding of the visual information displayed. Perception involves (1) vis-obj: extracting *visual information* (such as colour and shapes of objects) from the retinal image, (2) obj-prop: identifying the *semantic (propositional) relationships* from the information extracted (such as interpretation or identification of object boundaries, relationships of objects to other objects in the image), and (3) prop-implic: utilising these relationships in interpreting the *implicational meaning* relative to environmental contexts (such as ideational and affective content). There is a second route by which visual information can affect cognition (4) vis-implic: extracting

implicational meaning directly from the visual information. In addition to these bottom-up processes that take visual information to implicational meaning, there are top-down processes such as (5) implic-prop and prop-obj that interact to construct an object level representation from meaning or understanding.

A feature of the ICS model is the relation of inverse mapping (Herman and Duke 2001). This inverse mapping involves working backwards from the cognitive architecture of ICS to yield insights into desired structure and functionality of a rendering system that generates the image. These insights are helpful in deciding what type of information needs to be mapped in an image for better understanding and interpretation. For instance in ICS, for the propositional understanding of an image, there is a role of both object information and implicational information. This insight implies that both object information and implicational information needs to be mapped in an image for better understanding and interpretation. Therefore the structure of a rendering system should carry both object and implicational information for the designer to have a propositional understanding of the image.

Until recently, the drawings of cartoonists consisted of both the object as well as implicationally derived data such as sharp/threat, round/friendly paradigms. However, recent work on human animation, particularly in facial expressions, separated the basic geometry from the problem of configuring the geometry to capture expressions of mood, emotion, etc (Herman and Duke 2001). In such work, the separation of object information and implicational information is made *a priori* and coded into the rendering algorithm. Following this framework of separation are some works on non- photorealistic rendering (Hsu et al. 1993; Strothotte et al. 1994; Winkenbach and Salesin 1994; Meier 1996; Curtis et al. 1997; Litwinowicz 1997; Salisbury et al. 1997) that have concentrated on modifying the drawing attributes or rendering models to fit into this framework.

Duke et al. (2003) conducted a series of experiments to illustrate how rendering styles can convey implicational meaning and influence judgement, and how these effects can be controlled and used in a games environment. They have observed that on using rendering, a mode of processing is supported in which the more the implicational information contributes, the more likely that there is a clear propositional understanding of the image. The level of implicational information of the images influences cognitive processes such as interpretation, judgement, and imagination. This level pertains to the *emotive responses of the designer*. On the rendering side, or in terms of the computer graphics terminology:

- The implicational information of the images that stimulates designer's emotive responses can be termed as *emotive implications of computer graphics images.*

2.2. KANSEI ENGINEERING

Given its wide range of applicability, and in particular its treatment of human emotive responses, Kansei Engineering (Nagamachi 1995; Ishihara et al. 1997; Lee et al. 2000) was also chosen for this research. Generally Kansei, from a product perspective, refers to consumer's emotional feelings regarding a new product. These emotional feelings are usually based on the image of the product the consumers have in their minds (mental model) before buying the new product. Kansei Engineering is a methodology to translate consumer's feelings or perceptions on existing products or concepts into design solutions or design elements for new products. The standard procedure of Kansei Engineering involves:

1. Selection of adjective words for expressing the Kansei of the products, e.g. sporty, warm, cute, elegant.
2. Kansei Experiment: Evaluation of product samples using a semantic differential scale (Osgood et al. 1957) questionnaire.
   That is, the scales are established by determining the polar opposites for the selected adjective words and consequently placing them on the extreme ends of the scale (either 5-point or 7-point). The scale is then used to investigate the relevance of certain adjectives and its related design elements of the product.
3. Analysis of evaluation data: Factor analysis techniques are then used to identify those design elements that correlate with consumer's feelings. Thereby the designers can decide which design elements to adopt in the final product design.

3. Experimental Methodology

The objectives of the study were to (1) determine the emotive implications of images and (2) determine the effectiveness of different rendering styles in conceptual design.

The population of interest for the study were people who have had exposure to CAD systems. Owing to the difficulties of studying actual design practise in industry, a *convenience* sample (Howitt and Cramer 2005) of students and faculty, which represented the academia/education population, was chosen for the study. The sample comprised of fourth and fifth year undergraduate students, postgraduate students and faculty in the department of Design, Manufacturing and Engineering Management, University of Strathclyde. The students with architecture, computer aided engineering design, marine, and product design backgrounds had enough experience in design and carried out sketching and rendering activities as a part of their curriculum. The faculty had more than five years of experience in design. This acted as a basis that the findings are reflective to some degree of actual design activities. The study was conducted in a lecture room where distractions such as people walking through or talking has been excluded (Howitt and Cramer 2005).

The selection criteria of images for the study were based on models of images that were considered to be common and likely to be familiar (as it may be recognised easily as a particular model), available, able to be varied in different styles, and easy to render. To enable the respondents to perceive the images and get some feeling of a computer graphics rendering technique, the images were presented by printing them as high quality ink jet illustrations. The questionnaire comprised of semantic differential scaled response questions. A 5-point semantic differential scale ranging from +2 to – 2 was composed with polar opposites of the adjectives on the extreme ends of the scale, Figure 3. The respondents were also provided with a definition list, of the adjectives, to aid them while assessing the images. Having been presented with a selection of images, the respondents were asked to evaluate the image according to how much they felt a particular emotion applied to the rendering style. In essence they were asked to assess each image using the given adjectives that described the emotions.

	+2	+1	0	-1	-2	
Real						*Artificial*

Figure 3. Example of semantic differential type questionnaire.

(1) For the purpose of determining emotive implications, four images, one photorealistic and three non-photorealistic, were selected. Each image was rated against the same set of semantic differential scales. The objective of this kind of evaluation was to find out whether or not the images were perceived as eliciting the emotions described by the adjectives. This enabled us to assess the suitability of the adjectives to describe the emotive implications of images. The resulting list was used to identify the emotive implications of images.

(2) To determine the effectiveness of different rendering styles 30 images covering a spectrum of existing (a) photorealistic (PR), (b) non-photorealistic (NPR), and new (c) vague rendering (VR) styles, were selected[1]. The focus of PR styles was on images created using realistic *lights* (ambient, point, and spot), *materials* (normal, and transparent), and also *ray trace*. The focus of NPR styles was on *cartoon shading* (black and white, flat, gradient, hidden line, pencil sketch, colour pencil sketch, shadow, and shadow and highlight), *sketch rendering* (hand drawn, ink print, rough pencil soft pencil cartoon, colour wash, and line and shadow), and *technical illustration* (metal shading with edge lines, metal shading without edge lines, metal shading with hue shift, and phong shading) *styles*. The sketch rendering styles included the styles that were used for pen and ink illustrations, and painterly rendering. The focus of VR was on *line styles*, as a previous study on implications of rendering (Tenneti and Duffy 2005)

[1] Out of 30 images used in the study, examples of only a few are presented here, Figure 4, due to page restrictions.

established that vague rendering styles, especially lines, are of paramount importance in presenting vagueness. Therefore for VR styles a few possibilities of presenting vagueness were generated by using five line-types such as band-like, dotted lines with straight ends, sketchy, straight extended, and straight unextended lines, Figure 4.

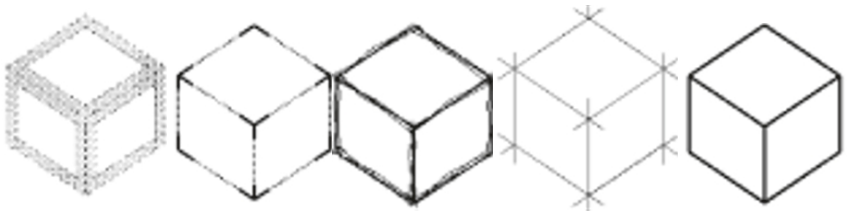


Figure 4. Examples of vague rendering line styles (adapted from Stevenson et al. 1996).

The study analysis consisted of quantitative compilation of the responses. The data collected from the semantic differential questionnaire were analysed statistically using the statistical program SPSS (Statistical Package for the Social Sciences) (Howitt and Cramer 2003). Consequently the findings were presented in the form of bar charts and tables.

The data in the bar chart are presented as summary statistics ± standard error mean. Where standard error mean is the standard deviation of the sample mean. Factor analysis was used to determine (1) the number of different factors needed to explain the pattern of relationships among variables used in the study, (2) the nature of the factors and (3) how well the hypothesised factors correlated with the different rendering styles. The factors extracted from the analysis are influenced to a greater or lesser extent by each of the original variables. The influence of each original variable on a factor is expressed as a loading coefficient. Loadings above 0.6 are usually considered 'high' and those below 0.4 are 'low'. The variables with high factor loadings are listed to see what patterns define the factor. Usually a cut-off point for factor loading is used to list variables under each factor generated. The variables with factor loadings greater than or equal to the cut-off point are then listed. The next step in factor analysis is to put a psychological interpretation on the emerged factors. That is, to decide the common theme in these high loading variables corresponding to each factor. At this stage either the factors emerged are recognised and named intuitively, or they are picked out as being in parallel with psychological explanations that were expected from theory or prior research (Coolican 2004).

4. Results

In total, 90 questionnaires were distributed to the sample and the percentage number of responses was 68% (61 returned responses). Out of the 61 responses, 33 (54%) were postgraduates, 22 (36%) were undergraduates and

6 (10%) were the staff. The respondents were from architecture, computer aided engineering design, marine, mechanical, and product design disciplines. The significant results extrapolated from the study are presented in this section.

4.1. EMOTIVE IMPLICATIONS OF IMAGES

The emotive implications of the images chosen for the study were the result of:

1. Explicit implications based on existing literature (Paterson 1986; Willows and Houghton 1987; Schumann 1996).
2. A study on implications of rendering (Tenneti and Duffy 2005).
3. Evaluation of implications derived from 1 and 2 by:
   - − Kansei Engineering semantic differential scales.
   - − Discussions with five experienced designers.

On scrutinising the implications identified from 1 and 2, for establishing the Kansei Engineering semantic differential scale, the following was noticed:

- − Some of the implications were adjective words themselves. And some others were jargon, which are the words or expressions developed for use within a particular group of people/context.
- − For such jargon (expressions), adjective words could not be derived to describe the implications.
- − Some of the implications were considered to convey the same meaning.
- − Some adjective words described two implications at the same time, e.g. *imaginative/creative.*
- − For some, polar opposites would not be meaningful or relevant.

Based on these deductions, eighteen adjectives describing the implications and their polar opposites were short-listed from those implications identified. Further evaluation by designers for the identified adjectives was performed to find out (1) how strongly they felt each of the adjective words apply and (2) whether or not the images were perceived as eliciting the emotions described using the eighteen adjectives. The designers were also asked to state any other specific implications of images that they felt were relevant.

Discussion with the designers revealed that (1) some adjectives used do not describe any implications of images; and (2) some adjectives used were describing two implications at the same time. Further, some other specific implications of images that the designers felt relevant were suggested. Based on their input, fourteen adjectives were finally identified, Table 1. These fourteen adjectives can be considered to describe the implications of images that stimulate designer's emotions.

TABLE 1. Fourteen emotive implications of images.

1. Approximation of information
2. Comfortable
3. Comprehensible
4. Differentiate of information
5. Exploration of alternate solutions
6. Imaginative
7. Interesting
8. Invites assumptions
9. Real
10. Recognisable
11. Satisfied
12. Stimulating to changes
13. Stimulating to discussion
14. Stimulating to look at

4.2. EFFECTIVENESS OF RENDERING STYLES

The questionnaire was designed to evaluate the effectiveness of Photorealistic (PR), Non-photorealistic (NPR) and Vague rendering (VR) styles. The respondent's ratings on the semantic differential scale indicate their perception and emotional responses to the images presented using these rendering styles. Unless and otherwise stated, the differences between the results obtained for the different rendering styles were statistically significant at the noted significant levels (either at 0.01 or 0.05).

The effectiveness of rendering styles is estimated in terms of it's 1) impact in conveying emotive implications and 2) correlation with emotive implications.

4.2.1. Evaluation using the fourteen emotive implications

The perception and emotional responses to PR, NPR and VR styles are presented in the form of a bar chart, as can be seen in Figure 5, where 4 in the bar chart indicates a strong positive response.

From Figure 5, PR styles have more impact in conveying all of the emotive implications (except *approximation of information* and *recognisable*) than NPR and VR styles. *Approximation of information* implies the ability of the images to display approximate or fuzzy or incomplete information of the design concepts. VR styles marginally convey this in a more meaningful way compared to NPR and PR styles. However, between NPR and PR styles, NPR illustrates the approximation of information slightly better than PR. *Recognisable* implies how well graphical objects in the image are identified. All the three rendering styles vary marginally, implying that the graphical objects in the image can be equally well identified using these three rendering styles.

Between NPR and VR styles there is a marginal variation for emotive implications such as *imaginative*, *interesting*, *satisfied*, *comprehensible*, *exploration of alternate solutions*, *invite assumptions*, *stimulating to changes*, and *stimulating to discussion*. This indicates that both these styles show more or less the same impact in conveying those emotive implications.

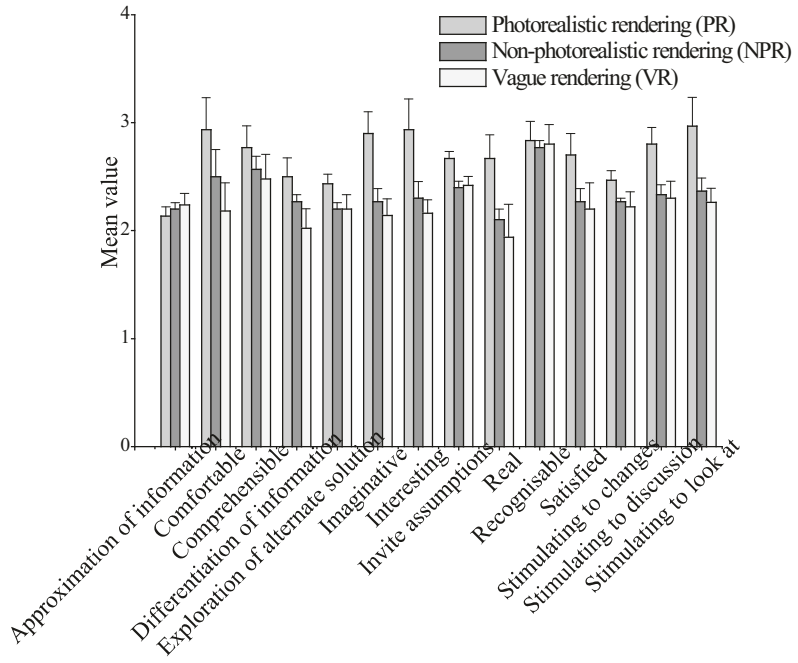


Figure 5. Impact of different rendering styles (mean values and standard error mean).

4.2.2. Evaluation using factor analysis

The evaluation of the rendering styles was further investigated by using factor analysis. Two basic analyses were conducted to firstly examine the pattern of relationships between the emotive implications identified (Coolican 2004) and secondly to establish the correlation between the different rendering styles and emotive implications.

From the first analysis, Table 2 shows the factors extracted and their respective factor loadings. Factor loadings are the correlation coefficients between the factors (columns) and the variables (rows). The strong correlations (factor loadings > 0.5) in the table illustrate that the fourteen emotive implications identified in the study deserve their place in the repertoire of the effects of images. It is clear from the table that four factors were extracted (columns). This shows that there were four underlying constructs that describe the emotive implications of the image. A cut-off point of 0.5 for factor loading was used to list variables under each factor

generated. The variables with factor loadings greater than or equal to 0.5 were listed under each factor.

TABLE 2. Relationships between the fourteen emotive implications of images.

	Factor			
Variable	**1**	**2**	**3**	**4**
Satisfied	0.8	0.0	0.2	0.2
Interesting	0.8	0.2	0.1	0.2
Imaginative	0.8	0.4	0.1	-0.1
Stimulating to look at	0.8	-0.3	0.3	0.1
Stimulating to changes	0.2	0.8	0.2	-0.1
Approximation of information	-0.1	0.7	0.1	0.2
Exploration of alternate solution	0.2	0.7	0.4	-0.2
Invite assumptions	0.1	0.6	0.5	0.2
Stimulating to discussion	0.2	0.2	0.8	-0.2
Real	0.0	0.1	0.7	0.5
Differentiation of information	0.2	0.2	0.6	0.1
Comfortable	0.4	0.3	0.5	0.1
Recognisable	0.1	0.1	0.1	0.9
Comprehensible	0.4	-0.1	0.0	0.8

The four groups are given different shadings in the table to show the variables constituting a particular group. The factors extracted have some interpretable meanings and a closer look at them in Table 2 reveals the following:

- Factor 1 is strongly influenced by the emotive implications of *satisfied*, *interesting*, *imaginative*, and *stimulating to look at*. Based on the nature of these emotive implications, this factor could be interpreted as an *affective* factor as it is concerned with intangible and abstract emotional aspects, like how interesting, or imaginative an image appears, or the experience of feeling satisfied with an image.
- Factor 2 is strongly influenced by *stimulating to changes*, *approximation of information*, *exploration of alternate solutions* and *invite assumptions*. This factor could be interpreted as a *functional* factor. As it is concerned with the functions to be performed for better presentation of the images.
- Factor 3 is strongly influenced by *stimulating to discussion*, *real*, *differentiation of information* and *comfortable*. This factor can be interpreted as a *motivational* factor. Because a motivational group describes to what extent the users are encouraged to participate in the design process.
- Factor 4 is strongly influenced by *recognisable* and *comprehensible*. This factor can be interpreted as a *cognitive* factor as a cognitive group is concerned with understandability of the displayed information.

The second factor analysis was carried out to establish the correlation between different rendering styles and factors extracted (that describe the emotive implications). This has been achieved by determining factor loadings for the 30 different rendering styles that represented PR, NPR, and VR styles. Figure 6 shows the average factor loadings of rendering styles.

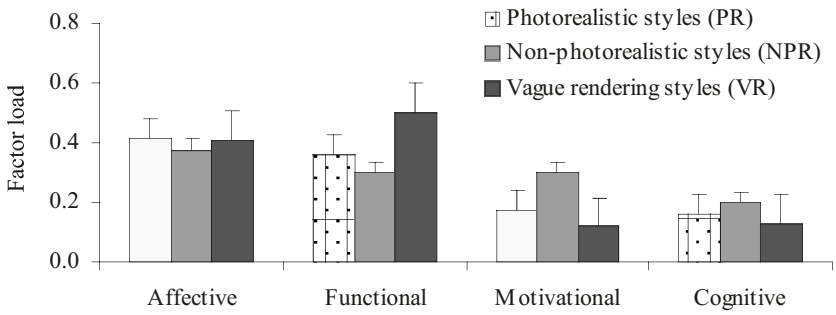


Figure 6. Correlations of different rendering styles and factors.

As can be seen from the bar chart, the factor loadings of PR and VR styles for *affective* factor is the same, indicating that both styles have the same correlation with the affective content. However there is a marginal variation between NPR, PR and VR styles, indicating that the correlation of all three styles in presenting the affective content is more or less the same. The factor loading of VR styles for *functional* factor is more compared to PR and NPR styles, indicating that VR styles are more correlated with the functional content. NPR styles seem to be more correlated, because of its high factor loadings, with *motivational* factor than PR and VR styles. Also NPR styles were conceived marginally more correlated with *cognitive* factor and making the content of the displayed information understandable than PR, followed by VR.

Based on the above observations, Table 3 presents the particular PR, NPR, and VR styles that correlate most with *affective*, *functional*, *motivational* and *cognitive* content. This has been achieved by identifying the particular PR, NPR, and VR styles with high *factor loadings* for affective, cognitive, functional, and motivational factors.

5. Discussion

Rendering provides graphical presentation of an image. This presentation can range from providing object information (whose precision varies from fairly vague to precise), emotive implications of images to evoking human emotive responses. The results suggest that the effectiveness of rendering styles can be evaluated by (1) Interacting Cognitive System (ICS) and (2) Kansei Engineering. Given its comprehensibility, the provision of a framework in which the operation of different micro-theories of cognition (such as bottom-up, top-down) can be situated and organised, and in

particular its treatment of two levels of information (propositional and implicational), ICS was chosen as the basis for our research. From an understanding of ICS, the significance of object and implicational level information for the human understanding or interpretation of the image generated using rendering has been distinguished. Also one important aspect in this processing of information is the interplay between bottom-up and top-down processing. This interplay is important in perception for two reasons. Firstly, the bottom-up processing in resolving ambiguities or uncertainties in the representations. And secondly, top-down in utilising the experience and knowledge of the user in forming an interpretation of images and also in generalising knowledge and matching patterns.

TABLE 3. Specific rendering styles to stimulate emotive implications.

	PR	NPR	VR
Affective	Spot light	Flat shading, Hand drawn	All vague rendering line styles
Functional			Straight extended lines
Motivational		Colour pencil sketch shading, Shadow and highlight shading, Hand drawn, Rough pencil, Soft pencil, Cartoons, Colour wash, Lines and shadows, Metal shading with edge lines	
Cognitive		Gradient shading, Shadow shading, Shadow and highlight shading	

The Kansei Engineering semantic differential approach adopted for the study proved to be a valuable methodology for the research in identifying the emotive implications of images and evaluating existing and new rendering styles. Central to Kansei is the viewer's opinion. Hence the identified emotive implications, and also the effectiveness of rendering styles, were the result of the viewer's perception and opinion. The semantic differential response questions proved to be successful as they facilitated quick completion and a detailed quantification of results. However there are some limitations such as:

- Respondents may have a tendency towards a 'position bias' where they habitually mark at the extreme end of the scale (or don't use

extreme ends at all) without considering possible stronger or weaker responses.

- There can be different interpretations of the ratings at the middle point of the scale. This could imply either the respondents do not have any of the subjective/emotional views or have neutral/indifferent subjective/emotional views of the images tested.
- The respondents felt tired evaluating thirty images in terms of fourteen emotive implications. There are chances for the quality of the response to be varied based on this.

The use of bar charts enabled comparison of the profiles of different rendering styles. On using factor analysis for studying the patterns of relationships among the fourteen emotive implications, four factors were extracted. Though naming these factors is a matter of subjectivity and possible dispute, based on prior research (Schumann 1996) and understanding in the field they were interpreted as *affective*, *functional, cognitive* and *motivational*. The variables with factor loadings greater than or equal to 0.5 were listed under each factor. However there will always be two or three variables that load an inappropriate factor and often will load into two or three different factors (George and Mallery 2005). Factor analysis does not prove that a real entity exists corresponding to each factor identified. It simply provides supporting evidence to claim that concepts could be arranged in a particular way.

It is intended that the results from the study will be used to improve a designer's experience with graphical images. The study suggests that the effectiveness of rendering styles can be evaluated by exploring the existing rendering styles such as (a) photorealistic (PR), (b) non-photorealistic (NPR), and new rendering style such as (c) vague rendering (VR). The effectiveness of rendering styles is estimated in terms of its impact and correlation. It is interesting to note that the impact of PR styles in *differentiation of information*, *exploration of alternate solutions*, *invite assumptions*, *stimulate to changes*, and *stimulate to discussion* is more than NPR and VR styles, Figure 5. These observations seem to differ from the normal interpretation of PR images as being more realistic and their value lies in representing the realistic appearance of real world environments and in the aesthetics of the resulting image.

It has been observed that *affective* content is equally correlated with all the three (PR, NPR, and VR) rendering styles, *functional* content more with VR styles, *cognitive* content more with NPR styles, and *motivational* content more correlated with NPR styles.

6. Conclusion

The paper presented a study of user perception and emotional responses to different rendered images using the Kansei Engineering to (1) determine the emotive implications of images and (2) determine the effectiveness of different rendering styles for conceptual design.

The study involved a sample of 61 students and faculty and 30 different rendering styles representing existing photorealistic (PR), non-photorealistic (NPR) and new vague rendering (VR) styles. Fourteen emotive implications were identified after evaluation of existing implications of images using Kansei Engineering semantic differential scales. The effectiveness of different rendering styles were determined by mapping the 30 different rendering styles with the identified fourteen emotive implications. The results from the study identified the following specific rendering styles (based on factor analysis) to stimulate emotive implications of images:

- *PR styles* such as *spot lights* are most effective for conveying *affective* content.
- *NPR styles* are most effective for conveying *affective, motivational,* and *cognitive* content. That is:
  - *Flat shading,* and *hand drawn* styles are most effective for conveying *affective* content.
  - *Colour pencil sketch shading, shadow and highlight shading, hand drawn, rough pencil, soft pencil, cartoons, colour wash, lines and shadows*, and *metal shading with edge line* styles for conveying *motivational* content.
  - *Gradient shading*, *shadow shading*, and *shadow and highlight shading* styles for conveying *cognitive* content, and
- *VR styles* are most effective for conveying *affective* and *functional* content:
  - *Band-like, dotted with straight ends, sketchy, straight extended,* and *straight unextended* line styles are most effective for conveying *affective* content.
  - *Straight extended* line styles are most effective for conveying *functional* content.

These results can act as a guide for designers to base their decision for using the respective rendering style/emotive implication paradigms in their drawings or for developing different presentations. They can also aid in development of images using the presentation styles that elicit the desired emotive responses. Eventually, the intention is that the image will be a better reflection of the designer's intentions (mental model) and thereby better support human-computer symbiosis.

It can be concluded from the study that vague rendering styles can be used in an effective manner by architects, graphic designers and other design practitioners for making graphical presentations accessible to changes, presenting approximate or incomplete information, stimulating exploration of alternate solutions, and making assumptions of an object in an image by using implicit cues. The study used line styles for the graphical presentation of vagueness. Since vague rendering had been established from the study as a useful form of image generation, other styles such as vague surfaces could possibly be considered for rendering vagueness.

Acknowledgements

The authors would like to thank the fourth year, fifth year undergraduate students, postgraduate studentsand faculty in the Department of Design, Manufacturing and Engineering Management, University of Strathclyde, who participated in the study, for their co-operation and also the University of Strathclyde for providing the studentship that enabled the research to be conducted.

References

Barnard, PJ and May, J: 1993, *Cognitive Modelling for User Requirements in Computers, Communication and Usability: Design issues, research and methods for integrated services*, Elsevier Science, Amsterdam, pp. 101-145.

Burns, KJ: 2001, Mental models of line drawings, *Perception* **30** (10): 1249-1261.

Coolican, H: 2004, *Research Methods and Statistics in Psychology*, Hodder and Stoughton, London.

Curtis, CJ, Anderson, SE, Seims, JE, Fleischer, KW, and Salesin, DH: 1997, Computer generated water colour, *Computer Graphics* **31**: 421-430.

Dooley, D and Cohen, MF: 1990, Automatic illustration of 3D geometric models: Surfaces, *IEEE Computer Graphics and Applications* **13** (2): 307-314.

Duke, DJ, Barnard, PJ, Duce DA, and May J: 1999, Syndetic modelling, *Human Computer Interaction* **13**(4): 93-158.

Duke, DJ, Barnard, PJ, Halper, N, and Mellin, M: 2003, Rendering and affect, *Computer Graphics Forum* **22**(3): 359-368.

Erwig, M and Schneider, M: 1997, Vague regions, *5th International Symposium on Advances in Spatial Databases (SSD'97)*, pp.

Ferwada, J: 2003, Three varieties of realism in Computer Graphics, *Proceedings of SPIE Human Vision and Electronic Imaging,* pp.

George, G and Mallery, P: 2005, *SPSS for Windows Step by Step: A Simple Guide and Reference 12.0 Update*, Pearson Education, New York.

Gooch, A, Gooch, B, Sherley, P, and Cohen, E: 1998, A non-photorealistic lighting model for automatic technical illustration, *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pp.

Guan, X, MacCallum, KJ, and Duffy, A: 1996, Classification of geometric design information and manipulation for vague geometric modelling, *Workshop on Knowledge Intensive CAD*, Carnegie Mellon University, Pittsburgh, USA, pp.

Guan, X, Duffy, A, and MacCallum KJ: 1997, Prototype system for supporting the incremental modelling of vague geometric configurations, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **11**: 287-310.

Halper, N, Mellin, M, Herrmann, CS, Linneweber, V, and Strothotte T: 2003, Towards an understanding of psychology of non-photorealistic rendering, *Proceedings of Workshop on Computational Visualistics, Media Informatics and Virtual Communities*, Wiesbaden: Deutscher Universitäts-Verlag, pp.

Heckbert, PS: 1990, Adaptive radiosity textures for bidirectional ray tracing, *Computer Graphics (ACM SIGGRAPH'90 Conference Proceedings),* New York **24**, pp.

Herman, I, and Duke, DJ: 2001, Minimal graphics, *IEEE Computer Graphics and Applications* **21** (6): 18-21.

Horvath, I, Rusak, Z, Vergeest, JSM, and Kuczogi, G: 2000, Vague modelling for conceptual design, *Proceedings of the Tools and Methods of Competitive Engineering*, Delft University Press, Delft, pp.

Howitt, D and Cramer, D: 2003, *A Guide to Computing with SPSS 11 for Windows*, Pearson Higher Education, Essex, England.

Howitt, D and Cramer, D: 2005, *Introduction to Research Methods in Psychology*, Pearson Education, Essex, England.

Hsu, SC, Lee, IHH, and Wiseman, NE: 1993, Skeletal strokes, *Proceedings of the 6th annual ACM Symposium on User Interface Software and Technology*, Atlanta, ACM Press, New York, USA, pp.

Ishihara, S, Ishihara, K, Nagamachi, M, and Matsubara, Y: 1997, An analysis of Kansei structure on shoes using self-organizing neural networks, *International Journal of Industrial Ergonomics* **19**: 93-104.

Kajiya, JT: 1986, The Rendering Equation, *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press, New York, USA, pp.

Kuczogi, G, Rusak, Z, and Horvath, I: 2000, Towards a natural user interface for comprehensive support of conceptual shape design, *UkrObraz2000 5th All-Ukrainian International Conference on Signal/Image Processing and Pattern Recognition,* pp.

Lee, S, Harada, A, and Stappers, PJ: 2000, Pleasure with products: Design based on Kansei, *Proceedings of the Pleasure-Based Human Factors seminar*, Taylor and Francis, Copenhagen, pp.

Lim, S: 2002, *An Approach to Design Sketch Modelling*, PhD Thesis, CAD Centre, University of Strathclyde, Glasgow.

Litwinowicz, P: 1997, Processing images and video for an impressionist effect, *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press, New York, pp.

McDonagh, D, Bruseberg, A, and Haslam, C: 2002, Visual evaluation: Exploring users' emotional relationships with products, *Applied ergonomics: Human Factors in Technology and Society* **33** (3): 237-246.

Meier, BM: 1996, Painterly rendering for animation, *International Conference on Computer Graphics and Interactive Techniques*, ACM Press, New York, pp. 477-484

Nagamachi, M: 1995, Kansei engineering: A new ergonomic consumer oriented technology for product development, *International Journal of Industrial Ergonomics* **15**: 3-11.

Osgood, CE, Suci, GJ, and Tannenbaum, PH: 1957, *The Measurement of Meaning*, University of Illinois Press, Urbana.

Paterson, G: 1986, An investigation of the presentation of graphical approximations, *Technical Note*, CAD Centre, University of Strathclyde, Glasgow.

Purcell, AT and Gero, JS: 1998, Drawings and the design process, *Design Studies* **19**: 389-430.

Rusak, Z, Horvath, I, Kuczogi, G, and Vergeest, JSM: 2000, Techniques for generating shape instances from domain distributed vague models, *Proceedings of UkrObraz*, pp.

Saito, T and Takahashi, T: 1990, Comprehensible rendering of 3-D shapes, *Computer Graphics* **24**(4): 197-206.

Salisbury, M, Wong, MT, Hughes, JF, and Salesin, DH: 1997, Orientable textures for image-based pen-and-ink Illustration, *Computer Graphics, SIGGRAPH'97 Proceedings Issue*, Addison Wesley, pp.

Schofield, S: 1994, *Non-photorealistic Rendering: A Critical Examination and Proposed System*, PhD Thesis, School of Art and Design, Middlesex University, United Kingdom.

Schumann, J, Strothotte,T, Raab, A, and Laser, S: 1996, Assessing the effect of non-photo realistic rendered images in CAD, *Proceedings of CHI'96 Conference on Human Factors in Computing Systems*, ACM Press, Vancouver, Canada, pp.

Sillion, FX: 1994, *Radiosity and Global Illumination*, Morgan Kaufmann Publishers San Francisco, CA, USA.

Smithers, T: 1998, Towards a knowledge level theory of design process, *Artificial Intelligence in Design*, pp. 3-21.

Stevenson, DA, Guan, X, MacCallum, KJ, and Duffy, A: 1996 Sketching on the back of the computational envelope…and then posting it? *AID'96 Workshop on Visual Presentation, Reasoning and Interaction in Design*, Stanford University, USA.

Strothotte, T, Preim, B, Raab, A, Schumann, J, and Forsey, DR: 1994, How to Render Frames and Influence People, *Computer Graphics Forum, Proceedings of EuroGraphics* **13**(3): 455-466.

Tenneti, R and Duffy, A: 2005, Identifying requirements for rendering in conceptual design, *International Conference on Engineering Design (ICED'05)*, Melbourne.

Willows, DM and Houghton, HA: 1987, *The Psychology of Illustration*, Berlin-Heidelberg-Springer-Verlag, New York.

Winkenbach, G and Salesin, DH: 1994, Computer generated pen-and-ink illustrations, *Computer Graphics, SIGGRAPH'94 Proceedings Issue* **28**(4): 91-100.

Yamaguchi, Y, Nakamura, H, and Kimura, F: 1992, Probabilistic solid modelling: A new approach for handling uncertain shapes, *Geometric Modelling for Product Realisation*, North Holland, Amsterdam, pp. 95-108.

IMPACT OF COLLABORATIVE VIRTUAL ENVIRONMENTS ON DESIGN BEHAVIOUR

MARY LOU MAHER, ZAFER BILDA AND LEMAN FIGEN GÜL
The University of Sydney, Australia

Abstract. A collaborative design environment makes assumptions about how the designers communicate and represent their design ideas. These assumptions, including the availability of sketching, 3D modelling, and walking around virtual worlds with avatars, effectively make some actions easier and others more difficult. An analysis of design behaviour in different virtual environments can highlight the impact and benefits of the different tools/environments and their assumptions. This paper reports on a study of three pairs of designers collaborating on design tasks of similar complexity using a different design environment for each task: face to face sketching, remote sketching, and 3D virtual world. Comparing the behaviour patterns and design actions we conclude that the characteristics of design process are quite different in sketching and 3D world environments. While sketching, the architects more frequently moved between the problem and solution spaces, dealing with analysis and synthesis of ideas. The same architects focused on synthesis of the objects, visually analysing the representation, and managing the tasks to model the design when they were in the 3D virtual world.

1. Introduction

Recent developments in networked 3D virtual worlds and the proliferation of high bandwidth communications technology have the potential to transform the nature of distance collaboration in professional design. There have been numerous developments in systems that support collaboration that have resulted in system architectures to support information sharing and remote communication. Whilst these initiatives have led to important advances in the enabling technologies required to support changes in global economic practices, there remains a gap in our understanding of the impact of the technologies on the working practices of the people who are the primary users of such systems.

Research into the characteristics of collaborative design can assist in our understanding of how the collaborative design process can be supported and

J.S. Gero (ed.), Design Computing and Cognition '06, 305–321.
© 2006 *Springer. Printed in the Netherlands.*

how new technologies can be introduced into the workplace. An understanding of collaborative design includes such factors as the role that communication media play, the use of physical materials and computer tools, and the way people communicate verbally and non verbally (Munkvold 2003). Protocol analysis has been accepted as a prevailing research technique allowing elucidation of design processes in designing (Cross et al. 1996). While earlier studies dealt mainly with protocols' verbal aspects (Akin 1986), later studies acknowledge the importance of design drawing (Akin and Lin 1995), associating it with design thinking which can be interpreted through verbal descriptions (Suwa and Tversky 1997; Suwa et al. 1998; Stempfle and Schaub 2002). By gathering information on how designers talk about and represent their design ideas during collaborative design while using different environments, we can understand how the characteristics of the different environments impact their focus during the design session.

2. Studying Team Collaboration

In order to understand the potential impact of high bandwidth environments on collaborative design, we first need to have data that characterizes collaborative design activity without the high bandwidth environment, i.e. face to face designing. We assume that the change in collaborative technologies should be incremental; moving from the technology already in use (usually sharing the drawings over the internet) to the use of a high bandwidth virtual environment. With these ideas in mind, an experimental study with 3 design settings was developed in order to study the impact of high bandwidth environments on design collaboration:

1. A collaborative design process in which designers work face to face with pen and paper.
2. A collaborative design process in which designers use a remote sketching system with synchronous voice and video conference.
3. A collaborative design process in which a 3D virtual world is used with synchronous voice and video conference.

We collected video and verbal protocol data in these three phases. Then we coded the behaviours we observed in the videos as well as the verbal communication. We analysed the codes and compared the collaborative activity in the three different settings, so that we could determine the impact of the change in collaborative technology. This paper presents the analysis of the data; comparing three pairs of architects' collaborative design processes in the three design environments: face to face sketching, remote sketching and 3D virtual worlds. The first collaborative environment represents the traditional way of designing, sketching, the second environment was selected as representative of the current low-bandwidth technology (Group

Board) and the third environment is a prototype of high-bandwidth technology (extended Active Worlds). The paper begins with a summary of the experiment design and then data collection methods. Finally, video and verbal protocol analysis of the design sessions and the results are presented.

3. Experiment

In our experiment, we studied pairs of designers collaborating on three different design tasks of similar complexity using a different setting for each task. We anticipate that the comparison of the same designers in three different environments would provide a better indication of the impact of the environment than using different designers and the same design task. Our designers are architects, so the design task is the design of a small building on a given site. We used the same site for each task, but specified a different type of building (gallery, library, and hostel) for each design task. This allowed the designers to become familiar with the site and to focus on the design of the building.

3.1. EXPERIMENTAL SET UP

Figure 1 shows the face to face session of the experiment where the designers are provided drawing materials (pen –paper), brief and a collage of the photos showing the existing building on the site and the neighbouring buildings.

Figure 1. Face to face session.

Figure 2 shows the set-up for the shared drawing board environment. In order to simulate high bandwidth audio and video, both designers are in the same room and can talk to each other, but can only see each other via a web cam. The set up for designer 1 is shown in Figure 2(a) and the set up for designer 2 is shown in Figure 2(b). The location of the cameras was an important issue, since we wanted to monitor the designers' movements, verbalizations, gestures and drawing actions. Cameras 1 and 2 capture the

gestures, general actions such as walking, looking at, moving to the side, while the direct connections to the computers/screens capture the drawing process. In this setting of the experiment, the designers used Group Board, as shown in Figure 3. One designer used a pen interface (Mimio) on a projection table, shown in Figure 2(a). The other designer used a pen interface on a Smart Board, shown in Figure 2(b).

(a) (b)

Figure 2. (a) Camera 1, Desktop screen 1, and Mimio on workbench; (b) Camera 2, desktop screen 2, and Smart Board.

In the third setting of the experiment, the designers used an extended 3D virtual world application in Active Worlds, shown in Figure 4. The 3D world includes a multi-user 3D building environment, video contact, a shared whiteboard, and an object viewer/insert feature. Again, the designers are in the same room with a similar camera set up. While the shared whiteboard was available in the third setting, the designers were only trained to use the 3D world.

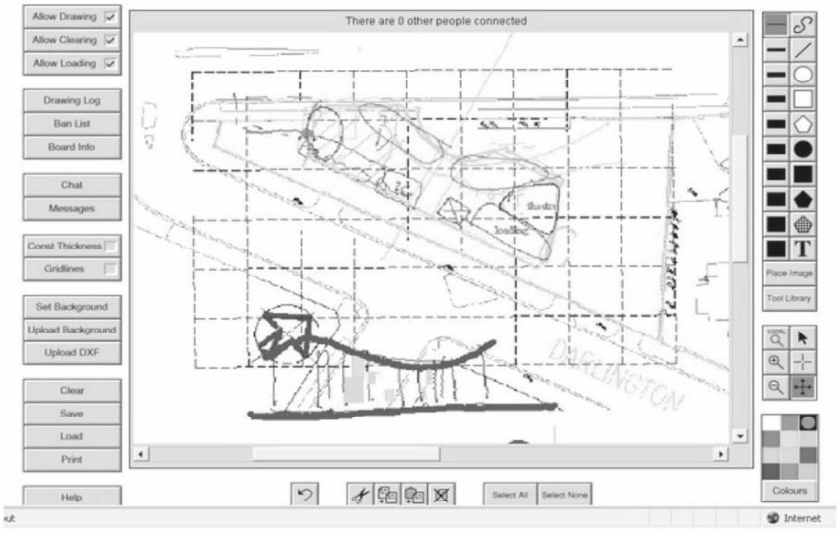


Figure 3. Group Board interface.

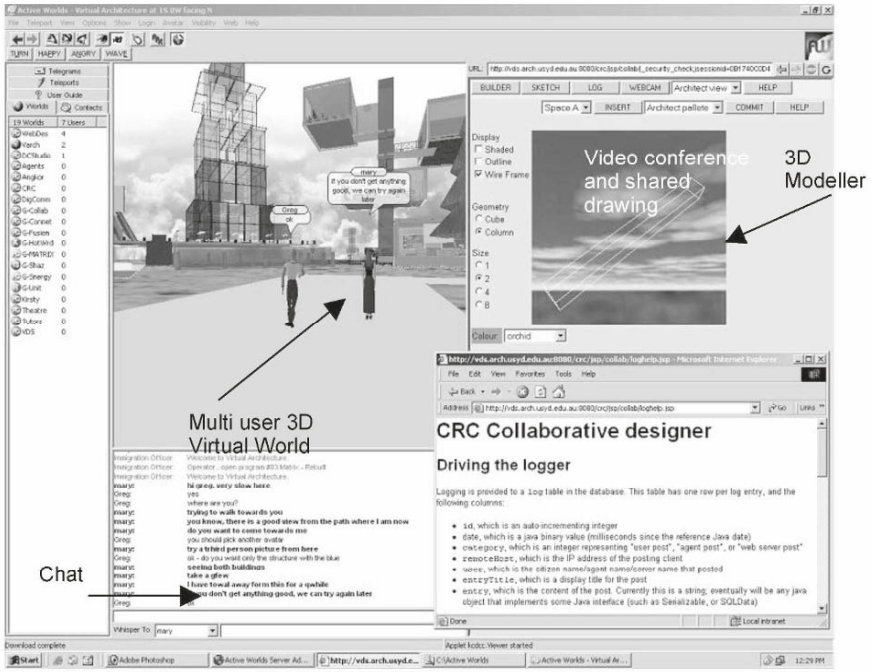


Figure 4. Extended virtual world.

3.2. EXPERIMENTAL PROCEDURE

The experimental procedure was:

1. The designers were given a design brief and a collage of the photos of the site they are required to build on. They were given time to read through the design brief and inspect the site layout and photos. They were given paper and pencils and were asked to complete their design session in 30 minutes.

2. The designers were presented a short description of how they could use the Smart Board and the Mimio Tool which both are pen and digital ink interfaces. The Smart Board is attached to a vertical plasma display and the Mimio is placed on a horizontal projection display, Figure 2.

3. The designers were given a 15 minute training session on the use of Group Board. In the training session participants were engaged in doing a tutorial in order to review and/or build their skills in using specific features of the application provided for collaboration.

4. The designers were given a new design brief and a collage of the photos of the same site. The site layout was set in the share whiteboard application as a background image on several pages so that the designers can

sketch on them. They were asked to complete their design session in 30 minutes.

5. After a 5 minute break, the designers were given a 30 minute training session on the use of extended 3D virtual world. Similar to the previous training session, they were asked to do a tutorial in order to review and/or build their skills in using specific features of the software application.

6. The designers were given a new design brief and a collage of the photos of the same site. This time the designers were using the extended 3D virtual world. They were asked to complete their design session in 30 minutes.

3.3. VIDEO AND VERBAL DATA CODING

The data from the experiments comprises 3 continuous streams of video and audio data for each pair of designers. In this paper we report on the analysis and interpretation of three pairs of designers. The stream of data for each session is segmented for coding and analysis. We used the software INTERACT[1] for our coding and analysis process; more information on the reasons for choosing this software and how it improved our coding process can be found in Candy et al. (2004).

Our segmentation is based on an interpretation of an event. In the study done by Dwarakanath and Blessing (1996), an event is defined as a time interval which begins when a new portion of information is mentioned or discussed, and ends when another new portion of information is raised. This event definition is an optimal one for our study as well, since the occurrences of actions and intentions change spontaneously as architects draw and communicate interactively.

An event can change when a different person starts speaking in a collaborative activity if s/he is introducing a new portion of information. In some cases the conversation goes on between the actors however the intention or subject of interest remains the same. In this paper we refer to the designers as Alex and Casey. For example, in Segment 48 both Casey and Alex take turns in one segment, however their subject of interest is still the "ramp to a car park":

Segment 48:

"Casey: This is... there is a photo of there. That is actually a ramp to a car park. And then there is a building and a little <inaudible>

Alex: And that is the ramp?

Casey: That is the ramp."

[1] www.mangold.de

3.3.1. Coding Scheme

Each segment is then coded according to a coding scheme. The coding scheme allows us to compare and measure the differences in the three design sessions. We used 4 categories of coding schemes: communication content, operations on external representations, design process and working modes.

TABLE 1. Coding Scheme.

Communication Content	
Software features	Software/ application features or how to use that feature
Designing	Conversations on concept development, design exploration, analysis-synthesis-evaluation
Awareness	Awareness of presence or actions of the other
Reps	Communicating a drawing/object to the other person
Context free	Conversations not related to the task
Operations on External Representations	
Create	Create a design element
Modify	Change object properties or transform
Move	Orientate/Rotate/ Move element
Erase	Erase or delete a design element
InspectBrief	Looking at, referring to the design brief
InspectReps	Looking at, attending to, referring to the representation
Design Process	
Propose	Propose a new idea/concept/ design solution
Clarify	Clarify meaning or a design problem, expand on a concept
AnSoln	Analyse a proposed design solution
AnReps	Analyse/ understand a design representation
AnProb	Analyse the problem space
Identify	Identify or describe constraints/ violations
Evaluate	Evaluate a (design) solution
SetUpGoal	Setting up a goal, planning the design actions
Question	Question / mention a design issue
Working Mode	
Meeting	Working together on the same design/artefact
Individual	Working individually on a different part/aspect of the design

The communication content category is applied to the transcribed conversation between the two designers, and one code is assigned to each segment. This code category has 5 codes as shown in Table 1. The communication on software features includes questions about how to do specific tasks with the software, talking about individual experience of how to do things, problems faced during the use of the software, any feedback about the interface or use of software /statements of frustration about not getting something right.

The operations on external representations category looks specifically at the actions the designers perform while using the environment (Table 1). Each segment is interpreted using the video of the designers' behaviour including movements or gestures, and the video stream of the computer display showing how the software was being used. Inspecting representations need further explanation because the action refers to different behaviour in 2D and 3D environments. Inspect representation in Group Board may refer to:

Looking at the representation and referring to its parts/aspect

Using hand gestures over the representation

Attending to a visual feature of the representation

Zooming in and out

Scanning

Inspect representation in 3D world may refer to:

Looking at the model and referring to a design object

Using hand gestures over the representation

Attending to a visual feature in the environment

Navigating or changing the view point in the environment

The design process category characterizes the kinds of design tasks the designers are engaged in for each segment, Table 1. Assigning a design process category takes into consideration the words spoken during each segment. The codes in the design process category are an adaptation of the coding scheme developed by Gero and McNeill (1998).

In developing the working modes category we took a similar approach to Kvan (2000) where he defined collaborative designing as a "closely coupled" process or a "loosely coupled" process. In a closely coupled process, designers work together on the same artefacts simultaneously while in a loosely coupled process, design participants work with different artefacts at a different or the same time. In this category "meeting" code refers to designers working together on the same design/artefact, and "individual" code refers to designers working individually on a different part/aspect of the design.

3.3.2. Combined codes

We combined some of the external representation codes and the design process codes into generic activity components in order to highlight observed different behaviours in the different environments. Create and Change activities represent the summary of operations on external representations. The design process codes are combined into four generic activities that are analyse, synthesize, manage task and visual analysis. By using combined codes we can more easily see the changes between the three design environments without getting lost in patterns of multiple codes. A summary of the combined codes is shown in Table 2.

TABLE 2. Combined Codes.

Combined Codes	**Individual Codes**
Create	Create
Change	Move, Modify
Analyse	Analyse problem, Clarify, Identify
Synthesize	Propose, Analyse solution
Visual Analysis	Analyse representation, Evaluation
Manage Tasks	Set up goal, Question

Create _Change

The 'create' operation in the FTF sketching environment was usually associated with drawing actions such as drawing a line, a complete or an incomplete shape, making symbols etc. Create action in Group Board sessions involved using drawing tools that are line, shape, fill etc. which is again similar to the FTF sketching. However Create action in the 3D world is usually just a click on an existing object, so that it is duplicated. Because the designers duplicate/create building blocks of space boxes, walls or columns, the building elements are created once and then re-arranged, by moving or modifying them. In our previous studies create - move - modify operations were observed to follow each other many times in the 3D virtual world and this pattern was associated to "making the model" (Maher et al. 2005a; 2005b).

Move (carrying an object to another position) and Modify (changing its properties) actions are related to either Group Board or 3D World environments. We combined move and modify actions under one name "Change", referring to the change in location or change of property of the entity or object.

Analyse_ Synthesize
We focused on two main activities that are related to the development of the design ideas at an abstract level. Analyze activity is assumed to take place in the problem space, and Synthesize activity is assumed to take place in the solution space. Analyze activity includes the following codes in the coding scheme: analyze problem, clarify, and identify. Synthesize activity includes the following codes: propose and analyze the design solution.

In protocol studies, analyze-synthesize activity refers to a design thinking cycle which involves analyzing a problem, proposing a (tentative) solution, analyzing the solution and finally evaluating it (Gero and McNeill 1998). A similar cyclic process was emphasized in creative cognition literature as explore-generate-evaluate actions (Finke et al. 1992). However in many cases, it is only after designers synthesize a solution that they are able to detect and understand important issues and requirements of the given problem. Lawson (1990) called this phenomenon "analysis through synthesis". The analysis of tentative solutions has been defined as a kind of design thinking and an expected behavior during the conceptual phase of designing.

Visual Analysis
Visual analysis activity is based on constructing a specific representation, thus the activity is different from analysis of the problem or the solution space. Analysis-synthesis refers to idea and design solution development via constructing an external representation. Visual analysis is purely dependent on the representation; judgments of what it should look like, how elements come together, designers' preferences on constructing it, and so on. Visual analysis involves seeing or imagining what the object looks like in 3D, so the "analyze representation" code is included in this activity. The evaluate code is included in this combined code as well because we observed that evaluation was mostly based on visual analysis.

Manage Tasks
Managing tasks refer to planning design actions ahead and leading the collaboration partner towards the goals to make the design. Questioning each other about design issues or knowledge is also involved in this activity. Manage tasks include the following codes from the coding scheme: Setting up a goal and questioning.

4. Interpretation and Discussion of Results

After coding each segment, the coding software INTERACT provides us with the total duration of each action in each category as well as how much time each participant spent on each action. The duration of each action is

divided by the total elapsed time for each session (which is 30 minutes for each session). This gives us the duration percentages for each action or action category. Table 3 shows duration percentages of the three action categories from the coding scheme. These are the averaged values of the three architect pairs collaborating in the three different design environments. Table 3 shows that around 72 percent of the total time is spent on collaborative communication in face to face sketching, Group Board and 3D world session. Thus the amount of communication is nearly the same in the three environments. The architect pairs spent 92-97 percent of the total design session time on operations related to external representations. Again the time spent on dealing with external representations does not seem to be significantly different over the three different design environments. However there is more variance between duration percentages of the design process actions category. In face to face sketching (FTF) session, architects spent 70 percent of their time on design process actions, however in 3D World session they spent 40 percent and in the Group Board (GB) session, 50 percent of the total time is spent on design process actions.

TABLE 3. Duration of action categories as a percentage of the total elapsed time.

	FTF	Group Board	3D World
Communication content	72%	73%	72%
Operations related to external representations	94%	92%	97%
Design process	69%	50%	41%

We tested if there are significant differences between the pairs in terms of their design behaviour (coded activity categories). The ANOVA test (ANOVA with replication, $P<0.05$) results show that there is no significant difference between the pairs' communication content ($p=0.58$), their operations related to external representations ($p=0.91$) or their working mode. This result supports that the architect pairs were similar in level of knowledge and experience, and their collaborative behaviour did not show a significant variance amongst the different pairs. Note that only design process is significantly different ($p=0.0015$) between the pairs. This result is not surprising since the design activity of one person might change due to the situations involved in the current context, and the variance in individual design strategies could have an effect on the collaborative design process.

It was also observed that the amount of time spent on communication in the three design sessions was very similar, Table 2; however the content of communication varied amongst the face to face sketching, remote sketching and 3D modelling environments. When we compared the communication content in the 3 environments, one significant difference was the amount of

communication about designing. This includes the design process related actions in our coding scheme, which could be interpreted as the actions needed for developing ideas/concepts and reasoning about them to reach a design solution. The ratio of talking about designing decreases from FTF to 3D world session, however percentages of other communication content categories increase. Figure 5 shows that the architects spent more time on the representation related context in the 3D virtual world. This involves talking about which elements they could use to represent their design ideas or what the representation looked like in the environment. The architects focused on the "representation" more in the 3D virtual world because they had to concretize their design ideas immediately, however in the sketching environment the representation could remain abstract.

Talking about software features occurred only in the digital media as expected, as well as the communication on awareness. Awareness percentages were higher in the 3D world. The discussion on awareness of others is due to the significance of the information about the other designer's location in the 3D virtual world and their actions with respect to the design model they are creating.

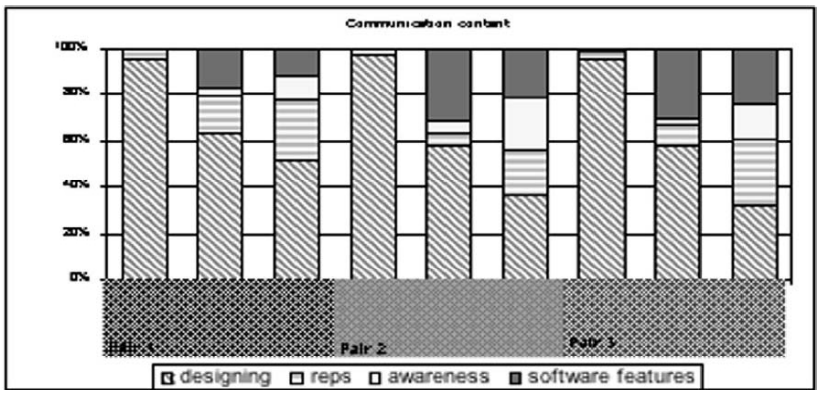


Figure 5. Communication content in face to face (FTF), Group Board (GB) and 3D World (3D) sessions for each pair.

Figure 6 shows the time spent on create and change activities which changed over the three environments. When we compare the Create and Change activities, one significant difference is that "Change" occurs in the remote/digital media, and the time spent on change activity is highest in the 3D world environment in all cases. Consequently Create activity has the smallest percentage in the 3D world, since the designers used the same objects by duplicating and moving them around. Thus the nature of the 3D modelling is not based on creating new things, as in the sketching

environments, where designers draw and trace over and re-draw the same things instead of copying them or moving them around.

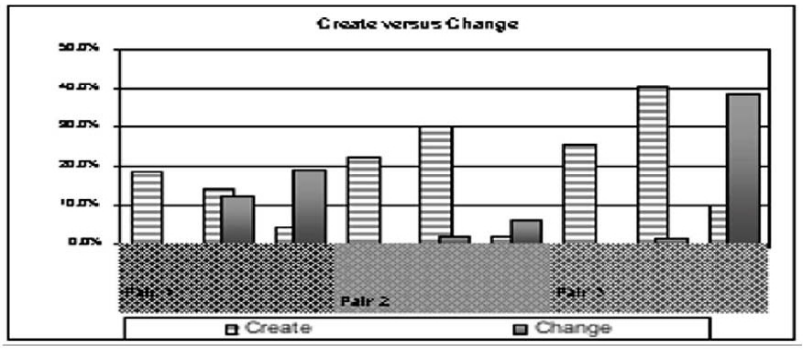


Figure 6. Occurrence of "Create" and "Change" activities in face to face (FTF), Group Board (GB) and 3D World (3D) sessions for each pair.

Operations on external representation category codes are shown along the timeline of the sessions, Figure 7. The beginning of the session is on the left, and the length of each horizontal bar indicates how long the designer spent on each operation. Each designer's external operations are coded separately indicated by the numbers 1 and 2. Figure 7 demonstrates Pair 3's external operations patterns visually in order to exemplify how we reached our conclusions about the action cycles. It can be observed in pair 3's actions chart that the FTF and remote sketching sessions have similar patterns in the operations on external representation and the 3D virtual world looks very different. In the FTF and remote sketching sessions the "inspect representation" was followed by "create" many times along the timeline, Figures 7(a) and 7(b). In the 3D virtual world the "inspect representation" was still followed by "create" and additionally followed by "move" and "modify" many times along the timeline of the session, Figure 7c. This demonstrates the relative richness of the 3D virtual world for manipulating the external representation.

Figure 8 shows the duration percentages of Analyse-Synthesize activity of the three pairs separately over the three design sessions. The graph demonstrates that there is a drop in the duration of the analysis-synthesis activities across the three design environments, FTF showing the highest percentages.

Figure 9 shows times spent on 'manage task' and 'visual analysis' activities in percentages. The graphs show that there is a significant increase in the duration of these design activities across the three design environments, 3D virtual world showing the highest percentages, Figures 9(a) and 9(b).

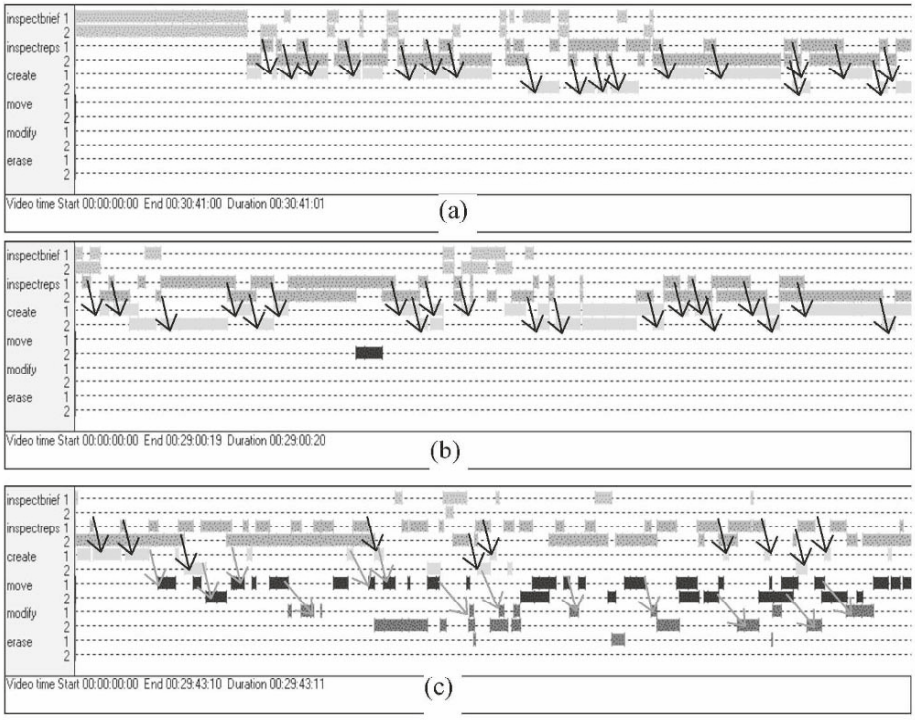


Figure 7. External operations of pair 3 in (a) Face to Face, (b) Group Board, (c) 3D World session.

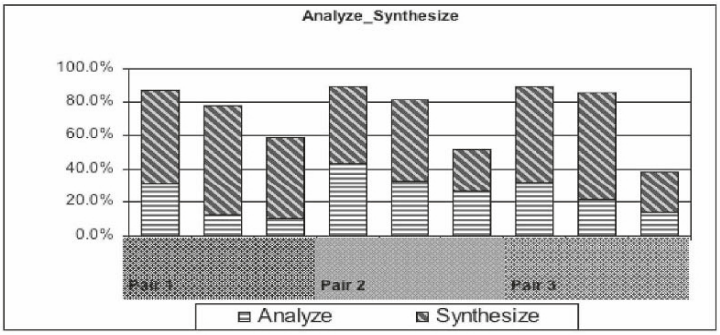


Figure 8. Analyse-Synthesize activity in FTF, GB and 3D World sessions for each pair.

A summary of our analysis of the working modes category is shown in Figure10. When the designers were working face to face, they were always engaged in "meeting" mode, during which they were communicating and acting on the same aspect of the design. When the designers were working

remotely, there was a small percentage of the time during which they were working on their own, focusing on different aspects of the design.

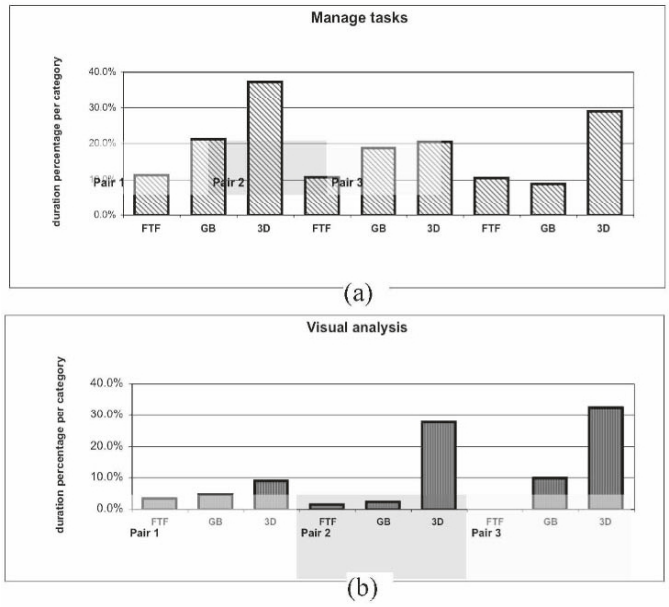


Figure 9. (a) Manage task (b) Visual analysis activities in FTF, GB and 3D World sessions for each pair.

For the three architect pairs' sessions analysed, the percentage of meeting working mode is highest for face to face and remote sketching sessions while the percentage of individual working mode is negligible. However in 3D world, architects worked less in meeting mode and relatively more in individual mode. This difference could be due to the nature of the 3D modelling environment, where participants have the opportunity to do task division and work separately (individual mode) on different aspects/parts of the design to be built. This result also shows that the 3D virtual world could support teams to work collaboratively but at the same time could support individuals to work separately in the different part/aspect of the design.

5. Conclusions

As available bandwidth increases and new virtual environments are developed to support collaborative design, designers are provided with a broader range of choices in how they communicate and collaborate at various stages of the design process. While it is essential and expected that the basic requirements for effective verbal communication are available during the collaborative session, there are numerous options for providing a shared representation of the design problems and solutions. In this study we

focused on the impact of moving from a familiar face to face sketching representation to two kinds of remote shared representation options: sketching on a shared drawing board and modeling 3D objects in a virtual world. Our study reports on 3 pairs of designers. While this is a small sample, the designers showed similar behaviors indicating that the results at least report on the kinds of differences and impact that we can expect to occur within the larger design profession. Our analysis, at a high level, shows that designers easily adapt to new environments as seen in our overall results on similar percentages of communication and operations on external representations. The difference in the environments is the impact of remotely communicated representations as sketches or 3D objects on the focus of the designers.

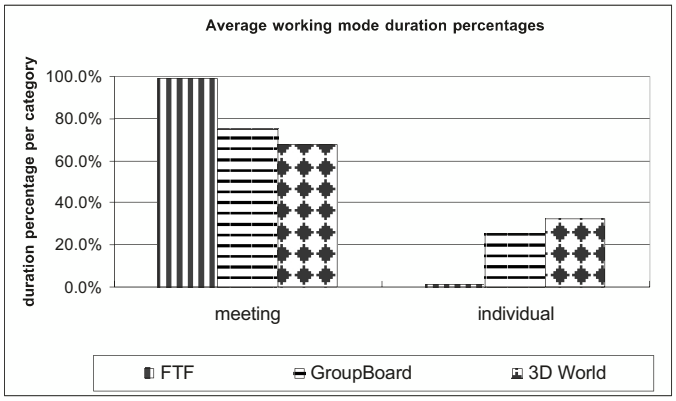


Figure 10. Bar charts for working mode of designers (average of 3 pairs over the 3 design phases).

The experiments described here characterize and compare the design behavior of pairs of architects using three different tools/media for designing. We demonstrated architects developed abstract concepts, analyzed, synthesized, and evaluated them when they were sketching and the same architects focused on synthesis and visual analysis of the objects and the making of the design, when they were in the 3D virtual world. Designers, while using the 3D virtual world and remote sketching, were able to move and change the objects and entities of their designs, allowing them to focus on iterations of the design solution. This is in comparison to face to face sketching in which a change meant redrawing the design representation. We also observed that the designers in the 3D virtual world spent relatively less time synthesizing design solutions when compared to sketching, indicating that the focus was on the design being modeled rather than generating numerous alternatives.

In conclusion, our studies show that while designers adapt to new environments and are able to effectively design face to face or remotely, the differences in the environments focus the designers on different aspects of the design process. Ideally, a designer should have multiple ways to communicate and represent the design problems and solutions. In our next set of experiments we provide the designers with sketching and 3D modeling within the same virtual world environment to determine how and when each is used when given the choice.

References

Akın, Ö: 1986, *Psychology of Architectural Design*, Pion, London.

Akın, Ö and Lin CC: 1995, Design protocol data and novel design decision, *Design Studies* **16**: 221-236.

Candy, L, Bilda, Z, Maher, ML and Gero, JS: 2004, Evaluating software support for video data capture and analysis in collaborative design studies, *Proceedings of QualIT04 Conference,* Brisbane, Australia (CD-Rom, no page numbers).

Cross, N, Christiaans H and Dorst K (eds): 1996, *Analysing Design Activity*, John Wiley, Chichester, UK.

Dwarakanath, S, and Blessing, L: 1996, The design process ingredients: A comparison between group and individual work, *in* N Cross, H Christiaans, and K Doorst (eds) *Analysing Design Activity*, John Wiley, Chicester, West Sussex, pp. 93-116.

Finke RA, Ward, TB and Smith, SM: 1992, *Creative Cognition: Theory, Research and Application*, MIT Press, Cambridge, MA.

Gero, JS and McNeill, TM: 1998, An approach to the analysis of design protocols, *Design Studies* **19**: 21-61.

Kvan, T: 2000, Collaborative design: What is it?, *Automation in Construction* **9**(4): 409-415.

Lawson, B: 1990, *How Designers Think*, London, Boston, Butterworth Architecture.

Maher, ML, Bilda, Z and Marchant, D: 2005a, Comparing collaborative design behaviour in remote sketching and 3D virtual worlds, *Proceedings of International Workshop on Human Behaviour in Designing,* Key Centre of Design Computing and Cognition, University of Sydney, pp. 3-26.

Maher, ML, Bilda, Z, Gül, LF and Marchant, D: 2005b, Studying design collaboration in virtual worlds, *in* JS Gero and N Bonnardel (eds) *Studying Designers'05*, Key Centre of Design Computing and Cognition, University of Sydney, pp. 291-305.

Munkvold, BE: 2003, *Implementing Collaboration Technologies in Industry: Case Examples and Lessons Learned*, Springer-Verlag, London Ltd.

Stempfle, J and Badke-Schaub P: 2002, Thinking in design teams- An analysis of team communication, *Design Studies* **23**: 473-496.

Suwa, M and Tversky, B: 1997, What do architects and students perceive in their design sketches? A protocol analysis, *Design Studies* **18**(4): 385-403.

Suwa, M, Purcell, T and Gero, JS: 1998, Macroscopic analysis of design processes based on a scheme for coding designers' cognitive actions, *Design Studies* **19**(4): 455-483.

DESIGN THINKING

Content-based Analysis of modes in design engineering
Pertti Saariluoma, Kalevi Nevala and Mikko Karvinen

Buildings and affordances
Alexander Koutamanis

The role of preconceptions in design: Some implications for the development of computational design tools
Patrick H T Janssen

How am I doing? The language of appraisal in design
Andy Dong

CONTENT-BASED ANALYSIS OF MODES IN DESIGN ENGINEERING

PERTTI SAARILUOMA
University of Jyväskylä, Finland

KALEVI NEVALA
University of Oulu, Finland

and

MIKKO KARVINEN
Technical Research Centre, Finland

Abstract. In this paper, we discuss about content-based design research. By means of it, we have separated four different types of thought processes, which occur during engineering design. We have used data which we have gathered during research into a large scale industrial design process, i.e., the design of the extended nip press (ENP) in paper machines. By means of interviews and documentary analyses, we have composed a picture of this complex design process. In it we have noticed that there are qualitatively different modes of thinking, which may be used to elaborate classical phase models of human thinking. We suggest that the human thought process entails such modes as apperception, restructuring, reflection and construction.

1. Introduction

1.1. CONTENT BASED DESIGN RESEARCH

Traditionally, the focus in engineering design research has been on special technologies and practices. Technical handbooks and practical engineering guidelines have been written and used for organizing design processes and thinking (Norman et al. 2000; Shigley and Mitchell 1990). In different ways, researchers have described what design engineers do, how they do it, how they have done it and how they should do it (Cross et al. 1996; Dym 1994 or Ehrlenspiel 2003; Pahl et al. 2003; Lindemann 2005; Suh 1990; Vincenti

J.S. Gero (ed.), Design Computing and Cognition '06, 325–344.
© 2006 *Springer. Printed in the Netherlands.*

1990), some focus on the interplay of the subconscious and explicit aspects of creativity (Altshuller 1999). It has been typical for these traditions that design thinking has been considered on rather intuitive and introspective grounds.

The rise of modern cognitive scientists has called attention to more systematic human research practices (Cross et al. 1996; Newell and Simon 1972; Simon 1969). Researchers have adopted a third person view and the objective methods of behavioral sciences to clarify the process of engineering design thinking (Akin 1980; Kavakli and Gero 2003; Visser 2003; Tversky 2003). These developments have added psychological concepts to the whole of multidisciplinary design research.

There are naturally several types of psychologically motivated approaches in design research. In this paper, we concentrate to develop the foundations of content-based design analysis. This means that we ground our argumentation in the information contents of designers' mental representations. This is why this kind of research can be called content-oriented of content-based (Nevala 2005a,b; Saariluoma 2002, 2003; Saariluoma and Maarttola 2001, 2003; Saariluoma, Nevala and Karvinen 2005a,b).

In content-based design research, we are interested in what types of mental contents may explain the essential aspects of design processes (Saariluoma 1984, 1990, 1992, 1995, 1997, 1998, 2001, 2002). Design is a typical problem for content-based design analysis. Problems such as the nature of the content structures in architectural design or in organizational thinking have been investigated (Saariluoma 2002, 2003; Saariluoma and Maarttola 2001, 2003; Saariluoma et al. 2004a, 2004b). The immediate roots of this work are in apperception research (Saariluoma 1990, 1995) However, its ultimate ground can be found in Wittgenstein's (1953, 1967, 1980) critique of Turing's (1936, 1950) computational model of mind. In his critique, Wittgenstein pointed out that Turing is not able to properly cope with thinking. Instead of formal approaches we should pay attention to meanings. The next natural step is to concentrate on the contents of mental representations, which explain meanings.

A natural problem for content based analysis is to investigate differences between various types of sub-processes in thinking. The literature thus indicates that design thinking is not a flat process, but it seems to have diverse stages or phases. Some parts of the process are qualitatively different from others (Dewey 1910; Newell and Simon 1972; Wallas 1926). Consequently, it is logical to ask, whether it is possible discern in design thinking different types of sub-processes.

2. Research Domain

Our investigation focuses on a very large scale industrial design process, namely paper machine design. This is an important subject for investigating design thinking as we do not have very much psychological knowledge about such processes. The artifact is very complex and the technical demands are high. The process of paper making includes a number of complex physical and chemical phenomena, and therefore, it is necessary that a reader with no previous experience on the subject has some idea about the key issues. For a detailed overview on papermaking technologies see Gullichsen and Paulapuro (1998-2000).

A paper machine is a huge construction of rotating machinery, pumps, pipe-work, driving systems and control equipment. It is composed of hundreds of thousands of separate parts. Its total weight is around five thousand tons. The whole line can be several hundreds meters long – the paper machine itself extends over hundred meters. All this is put together for the purpose of removing water from a liquid solution of about 1 % fiber content in order to form a paper web with an optimal moisture percentage for further processing, and meanwhile conveying the web about hundred meters by a speed of nearly 100 km/h. The final thickness for example of newsprint paper is about 0.07 mm and the web can be over 10 meters wide. Understanding the design of these machines is our goal.

To make our task simpler we have concentrated on the 20 years long design process of extended nip press (ENP) for paper machines, Figure 1.

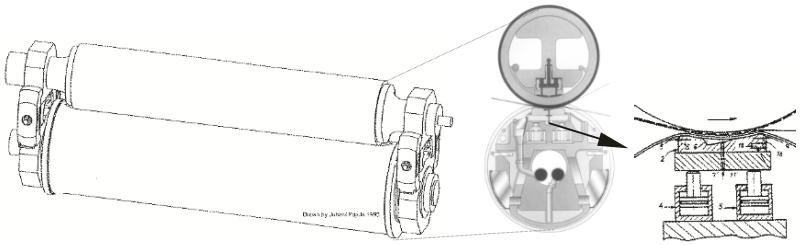


Figure 1. An extended nip press (ENP) for paper machines. It provides a wider contact zone (i.e. the nip) between two rolls and consequently a longer press impulse on the fast running paper. The lower roll has a flexible mantle, which is pressed by the upper roll against a contoured "press shoe" inside the lower roll. Sources: Patent publications (Finland 1995; PCT 1993a) and the publication of 10th Valmet Paper Machine Days (1996).

3. Method

3.1. GENERAL METHOD

Online monitoring of a long range design engineering thought process is not possible. Therefore, we adopted a reconstructive approach, where we

utilized individual and group interviews and document analysis to reconstruct a two decades long creative thought process (Nevala 2005a, 2005b; Saariluoma et al. 2005a, 2005b). This kind of analysis is qualitative, which is natural as we are interested in the contents of the thoughts (for qualitative analysis see Dey 1993 and Denzin and Lincoln 1994; Hodder 1994; Smith 1994). A classic example in the psychology of thinking is provided by Max Wertheimer (1945) in his classic study on productive thinking. He systematically interviewed Einstein in order to understand the thought process, which led to the theory of relativity.

In this kind of methodology, it is impossible to document minute by minute or day by day what happens in the minds of several people. Nevertheless, we can reconstruct a history of thoughts and ideas and use methods typical to qualitative analysis of interviews and documents. In this way, we achieve new types of information about design thinking.

3.2. HOW THE RESEARCH WAS CARRIED OUT

The empirical study was carried out in 2003 – 2005 at Metso Paper Inc. Rautpohja Works, in Jyväskylä Finland. We had possibilities to interview five engineers, who were centrally involved in the development process in diverse organizational positions.

We started the interviews with group meetings, which all participants attended. On the basis of these meetings and related documentary material we established an overall reconstruction of the development process, Figure 2.

Starting from June 2003 the individual interviews were performed and more documented material was collected. We had the possibility to repeat the interviews, when necessary for completing the information. The engineers were interviewed up to four times. The interviews were complemented by telephone discussions and email inquiries.

The obtained material includes 17 hours 50 minutes of individual and group interviews, large assortments of related organizational documents and nearly a hundred patent publications.

Data analysis: In data analysis for reconstructive research, we have separated the indicative elements from the total mass of data (cf. Duncker 1945; Ericsson and Simon 1980, 1984; de Groot 1965; Saariluoma 1984, 1990, 1995; Wertheimer 1945). By an indicative element, we mean observational elements which may have an argumentative value. After that we describe and classify them. On the grounds of the analysis received, the actual argumentation is presented.

In our analysis, we do not directly use what the engineers say as evidence, but we have focused on the mental representations behind the linguistic information.

4. Results

4.2. THE FOUR MODES OF THINKING

The general overview of the development process can be found in our earlier papers (Saariluoma et al. 2005a, 2005b). We have built our argumentation on observations available in the collected empirical material. Our focus is on searching for and defining qualitatively different modes of design thinking. Naturally, this kind of explicative approach has three stages. Firstly, we have to discern a target phenomenon, i.e., a mode of thinking, secondly, we have to conceptually describe it and explain how it differs from the other modes and conceptualize it, and finally we have to name it in order to submit it under scientific discourse. This procedure can be called explicating (Saariluoma 1997). In our analysis we found evidence for four qualitatively different modes. We termed them apperception, restructuring, reflection and construction.

4.2.1. Apperception

In earlier content-based research, on thinking, the difference between perception and the construction of mental representations has become evident (Saariluoma 1990, 1992). This is a classic difference, which was originally made by Leibniz (1704) and Kant (1781) among others (Stout 1896; Wundt 1880). The content-analysis of mental representations during various thought processes has shown that people incorporate non-perceivable kinds in the content of representations. Typical examples are such notions as possible, infinite, tomorrow; force, mass, friction and electrons. This means that the origin of mental contents cannot be solely in perceptual processes. The first thing to look at is whether this distinction has a role to lay in engineering design thinking.

EXAMPLE 1: The need for making use of the previously known idea of an extended nip zone in the wet-press was connected to the strategic decisions of considerably increasing the running speed of paper machines.

(T10, 00:09:51[1]): *... On the other hand, at that time the speed was increased. Actually that was why it [the extended nip press] became interesting... When I came here in 1979, I came to the project for increasing the speed... The idea came from outside Valmet... It lead to the approach of increasing the speed, in other words we saw the problems in increasing the speed and tried to search for the solutions... There was the question of supporting the paper web, but at the same time it was seen that when increasing the speed the dry content [of paper] goes down. So increasing the pressing time was a natural idea.*

[1] Position on the CD of the interview recordings: track 10, 00 h, 09 min, 51 sec.

Explanation: In 1970s, in the papermaking industry, a search for technology, which could make it possible to significantly increase the running speed of the paper machines, had been started. One of the many ideas involved, which the product development engineers created, included the possibility of extending the press zone of one press nip.

Later on, the importance of the extended nip was reinforced in the minds of the engineers by the delivery of a "shoe press" into a board machine by the American Beloit Corporation in 1981 (Justus and Cronin 1982).

In the example, the designer concentrates on some aspect of the manifold of possibilities. Evidently, there is very little perceivable in this representation. The plan was to make a machine, which allows doubling the production speed. This representation must have been fully conceptual. For this reason, it is logical to use the notion of apperception. The example also illustrates how apperception focuses on the contents of mental representations.

EXAMPLE 2: One problem inflicted by the need for increasing the machine speed was the elimination of vibrations. At some stage in design problem solving, a designer found a potential solution; a way of thinking the whole. In this point apperception constructs a consistent mental representation.

(T4, 00:13:48): *... Here is the ZS roll. The bearing houses are pressed by a load cylinder together. And then the roll mantle is moving… the load is applied on the mantle. Inside the roll there are a row of load cylinders… a bit like in the belt roll [the extended nip roll]. We got off the vibration problem by this…*

Explanation: The development of the SymZS-roll was one example of the diverse consequences of the increased running speed of paper machines. The vibrations were controlled and an even the line pressure of a long roll was possible to adjust by this innovation.

Here, we can see another characteristic of apperceived mental representations in paper machine design. It pre-supposes the joining of various content elements such as cylinder, mantle and load. However, we learn here that apperception integrates various content elements as a whole.

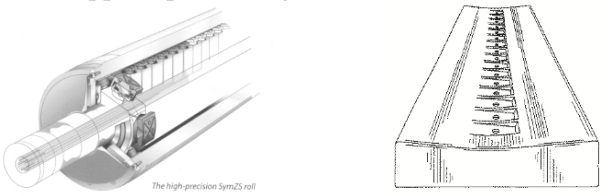


Figure 2. Pressure pockets of the SymZ-roll were transformed to hydrostatic pressure pockets of the so called hybrid (i.e. hydrostatic-hydrodynamic) press shoe.

EXAMPLE 3: The content elements of mental representations in design are normally integrated into sense making or senseful wholes (Saariluoma 1990, 1995). In the current interviews, this character of mental representations was explicit. The systems of reasons could be constantly seen. A good example is presented in Figure 3. It concerns the mental representations of the hydrostatic pressure pockets.

Explanation: The main reason for inventing the pressure pockets was to decrease the friction of the press shoe, but it also gave the possibility of adjusting the pressure conditions, i.e., the pressure curve in the direction of the run of paper web.

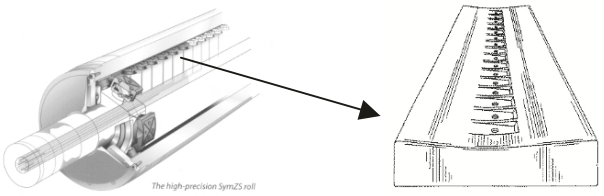


Figure 3. Pressure pockets of the SymZ-roll were transformed to hydrostatic pressure pockets of the so called hybrid (i.e. hydrostatic-hydrodynamic) press shoe.

(T8, 00:24:26): *... This is good in that way that there is no friction here. It is on the oil cushion… The nip [pressure] curve in a hydrodynamic press shoe always obeys the physical laws and becomes this kind… We thought that it would be convenient if the nip curve could be adjustable… These were the two reasons…*

Here, a number of interconnected reasons such as friction, nip pressure curve make the structure of the representation understandable. One can say that these reasons explain why we have precisely the system of elements in Figure 3 as we have. Apperception constructs thus mental representations with interconnected and sense making elements (Saariluoma 1990; Saariluoma and Hohlfeld 1994; Saariluoma and Maarttola 2003).

Discussion

In the presented cases, the existence and the major conceptual criteria for the apperception mode are explicable and they give several types of information about the apperception mode. Firstly, the thought processes behind these schemes entail much non-perceivable information such as hydrostatic and hydrodynamic pressure or friction. Secondly, they define the temporarily prevailing system of pre-suppositions. This means that designers focus on some particular type of mental contents. This content entails the hypothetical "idea" which is tested and validity inspected. These are naturally also solution attempts. An important property is that the prevailing representation blocks out alternative representations for the time of processing the selected one (Saariluoma 1995). Bartlett (1958) described this phenomenon by the

term "point of no return". In addition, apperception organizes representations as functional, senseful and coherent combination of attributes. These properties have an important role in the structure of mental contents (Saariluoma 1995 for a similar analysis of apperception within a different empirical domain).

One may also notice that there are two stages in apperception. The first is fast and subconscious. We can term it dynamic apperception. It is the process in which the representation is formed. Second stage is stabile, and in this stage mental representation comprises all its structural properties and it is also conscious as far as possible. Apperception is a mode in which people have an idea about what to do. However, they do not necessarily know that it is the right thing to do and thus the solution is compared to reality.

The self-consistent structures of representations are a major characteristic of the apperception mode. This means that there is a reason for each element for being incorporated in a representation. This property gives mental representations their sense-making or senseful structure (Saariluoma 1995; Saariluoma and Maarttola 2003).

4.2.2. Restructuring

In an empirical analysis, the intuitively emerged apperceptive representation seems to fluctuate and change. These changes from one apperceived representation to another can be called restructuring (Köhler 1930, 1956; Wertheimer 1945). Restructuring the mental representation is triggered by the information, which is considered inconsistent in the apperceptive representation. This means that the subject sees it impossible to reach the goal.

EXAMPLE 4: The quotation in CASE 1 one continues as follows:

At the same time it was seen that putting nip after nip would be a rather expensive solution, but if it were possible to handle the situation by lengthening the nip, so why not...

Obviously, the engineers were dissatisfied with the natural idea of increasing the number of nips to increase the press impulse. Consequently, they rejected this idea and searched for a new, Figure 4. They restructured their mental representations as the solution would not work.

EXAMPLE 5: The reason for restructuring can be the falsifying information induced by the professional knowledge of the designer and the comparison of a hypothetical solution to the problem situation.

(T4, 00:02:50): *... Here again… two static pressure pockets. This is one of these propositions of the belt extorted press nip constructions [which do not work in reality]…*

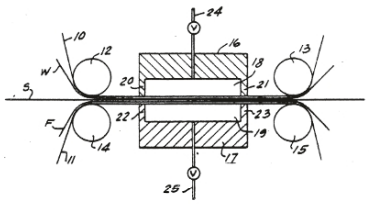


Figure 4. Extended press nip according to patent publication USA (1974).

Explanation: The engineer could conceive without calculation that it is not physically possible to achieve the pressure necessary for an extended press nip by the means of two belts and two pressure pockets only. Mechanical means in maintaining the pressure is also needed. Consequently, all these kinds of constructions were abandoned and more consistent representations were sought. This made it necessary to restructure the prevailing representation. It was put aside and a new apperceptive process constructed a new representation.

EXAMPLE 6: According to the interviews the general policy of product development at Valmet Paper Machines at the beginning of 1980s relied largely on the survey of new inventions elsewhere. In this kind of product development strategy, it was natural that the prevailing representations studied in other manufacturers' plans were systematically redesigned. The situation naturally required fast restructuring processes from the designers. The request was to create better solutions from technical and marketing points of view.

(e.g., T1, 00:01:38): *... It was the good old strategy of being "the best second" in the markets. ...Nothing advanced in the organization if the competitor had not introduced it first. The customers simply do not ask for it, even if it could be a superior solution…* (Figure 5).

Explanation: Product development engineers were obliged to restructure constantly. New inventions and new patent publications were looked at in the mode of restructuring them "to a better invention which did not violate the initial patent"

(T16, 00:01:15): *... Beloit's shoe press was known by us and the patents of it. And also the Voigth's closed model was known… There was such an enormous jungle of patents that we thought whether there would be room for another one at all... We went on thinking that we had to find a better invention so that it would be sales attractive, and could be patented. We found out that Beloit had patented a hydrodynamic press shoe with a single joint… which determines the operation of the hydrodynamic bearing… We thought that we will make a virtual joint by two load*

cylinders, where we will get a virtual joint, which is not seen, by adjusting the pressures of the cylinders…

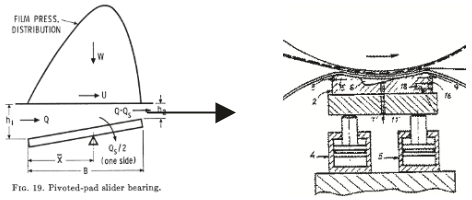


Figure 5. Result of the intentional restructuring mode of thinking of the designing engineer, i.e. a virtual joint, constructed by two load cylinders (b) instead of one mechanical joint (a).

Explanation: The inconsistency which caused the restructuring of the mental representation in the engineer's thinking (illustrated by Figure 5a) was the patent protection already established by the Beloit Corporation. The familiar principle of the hydrodynamic bearing did not suffice. It was necessary to restructure the mental representation.

Discussion

In the presented cases, designers find reasons why the ideas do not work in reality. This is an essential difference between apperception and the next mode we call restructuring. In restructuring, we can find both elements about the possible solution and also information which makes it impossible to utilize the prevailing interpretation. Normally, the focus moves from the solution candidate to the falsifying reason.

Consequently, people abandon the prevailing representations. Falsifying the information makes it impossible to continue in the original direction and therefore designers have to change their way of seeing things. They restructure their representations. In this mode, the content changes, designers move to another representational content and return anew to the apperceptive mode. Basically restructuring is a shift between two apperceptive processes.

In restructuring one can also see two stages. Firstly, one can find the static? stage, in which the focus of thoughts moves? to the crucial inconsistency, and secondly, a fast and subconscious stage of dismantling the prevailing mental representations and replacing it with a new one. The latter stage is fast and we have very little observational knowledge about it.

Restructuring is a qualitatively different mode from apperception. One could characterize restructuring as the process of realizing the fatal inconsistency in the prevailing representation (hypothesis), abandoning it and starting the construction of a new representation. When in apperception we are interested in the way functional reasons bind the elements of the structure into a coherent whole, the major characteristics of restructuring is

some inconsistency, which makes the hypothetical solution impossible to realize or non-purposeful to apply (e.g., for economical reasons).

In this vein, restructuring is negative compared to the positive characteristic of apperception. They are qualitatively opposite but complementary modes. Apperception leads to restructuring and restructuring to apperception but they do not prevail at the same time. Their mutual "dance" or fluctuation forms the basic dynamic movement of thought.

4.2.3. Reflection

When we follow the development of design process as a whole, it is possible to find a new mode of thinking. Clearly, designers have several hypotheses, which are mutually inconsistent in content. It is possible to think, for example, that the flexible belt is either of fiber reinforced polyurethane or of metal, but it is impossible to think of it both at the same time. The shifts in apperception and restructuring seem to lead to a higher level mode in thinking: moving from one possible solution to another.

De Groot (1965), for example, noticed that chess players may move from one hypothesis to another and return to some earlier alternative but in an improved form. He called this progressive deepening. Saariluoma (1995) pointed out that the movement from one problem sub-space (or a solution possibility) to another extends the search beyond a single sub-space (or solution possibility). This means that we should look at the problem solving process from a wider perspective than simply alternating between apperception and restructuring.

This means that we should perhaps think of a new mode, which entails comparisons between alternative hypothesis and solution candidates. The chains of patents provide us with very clear evidence of the existence of such a mode. We termed it reflection, because it is obviously very close to considering, pondering and other everyday reflective thought processes.

EXAMPLE 7: Figure 6 illustrates a set of alternative apperceptive representations in how it would be possible to connect the flexible belt to the end plates of the roll in a reliable way. The restructuring modes follow each other. The designer's thinking moves from one way of solving a problem to another. This series of alternatives, picked up from a patent publication, documents the movements of thoughts of the design engineer. In this case, alternatives are mutually exclusive. This means that only one of them can be applied.

EXAMPLE 8: Another kind of example of the reflective mode of thinking is from the interview of engineer D. This example shows that one change in the operating parameters of the paper machine (a result of a bunch of completed

thought cycles elsewhere) normally leads to a chain of reflections. This example also provides a sample of our analysis of the interview protocols and shows clearly how the modes of thinking are entangled with each other.

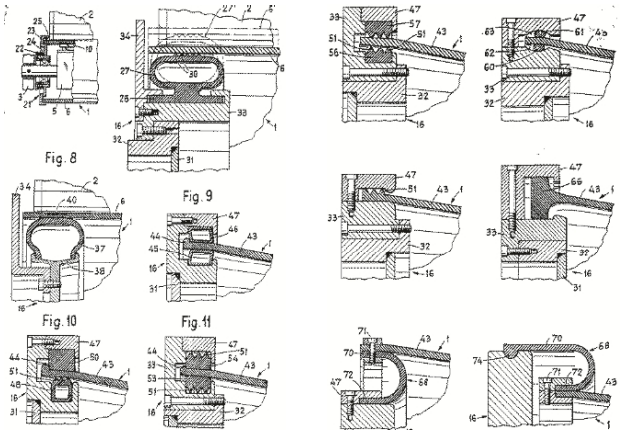


Figure 6. Alternative designs for mantle-end plate fastening in the flexible roll of an ENP-unit. According to patent publication (Germany 1985).

(T4: 00:10:00): ... *When the speed of the paper machine was increased, we got a vibration problem between the center roll and the grooved press roll…*

Explanation: The designer had obtained a representation of the consequences in the change of speed. The inconsistencies of the representation induced a restructuring process in the designer's mental representation of the operating conditions in the press nip. This triggers the need for reflective thinking.

... We then started to think what should be done. In the third press, there was a single-felt nip and the felt was rather thin. It readily popped to mind that a thicker felt would dampen down the vibrations. An additional advantage would be better water removal… But there were difficulties in the felt changing procedures and equipment… Years later the vibration damping was managed by replacing the SymZ roll by SymZS roll…

Explanation: The reflective mode of thinking led the designer to search for a more consistent representation. In his mental representation the inconsistency was the single thin press felt, which led to the thought about thicker felt. This caused inconsistencies elsewhere, which needed further reflective thinking. The emergence of the SymZS roll shows that in a large organization the solution to a problem can emerge after a large innovation cycle as a confluence of innumerable constructive thought cycles.

Discussion

The examples illustrate how reflection works above or over the apperception-restructuring-cycle. It is a pre-supposition for human-like problem solving in which comparisons between alternative "ideas" are normal. Apperception and restructuring generate new alternatives but reflection is required to select among alternatives and improve them.

The mutually contradictory content of alternative solutions makes it necessary to postulate that mental representations controlling the alternation of apperceptive representations must be controlled on meta-level. The process of alternating between various solution models is called reflection; the reflective mode of thinking. It may have contradictory content elements and this makes it qualitatively different from apperceptive, restructuring and constructive modes.

The crucial difference between reflective, apperceptive and restructuring modes can be found in the focus. Apperception and restructuring focus on one single unified representation. Reflection compares between several alternatives, which are essentially incompatible with each other. This is why it is a conceptually different mode than the first two and can be called reflection. Perhaps, the dominant characteristic of reflective thinking is where to go in the "forest" of possibilities. It is essentially the selection between various possible ways of acting.

4.2.4. Construction

Finally, our results suggested that design engineering thinking has a constructive character. This means that element after element is incorporated into the final design. The elements are combined and included in a harmonious whole. Analysis of our research material shows how the designers construct the whole; step-by-step and problem-by-problem.

CASE 9: (T4, 00:14:44) ... *We got thicker felts and ZS-roll… So we had to do something to the third press in order get the felt change in order…*

Again (T4, 00:16:10) ... *I think that it was some machine in Rauma, where we built the first cantilever press section…*

(T4, 00:21:35): ... *[Cantilevered press means that] this felt can be pushed as a tube into the press ... The felt can be as thick as needed…*

(T4, 00:23:35): ... *Then the latest one is the jointed felts. ... Damn simple frames… ... the felt is not an endless loop, the ends will be joined together… The new felt will be drawn in by ropes ...and somewhere here joined. It can be used also in old machines, when thicker felt is needed. ... Nowadays the joint is so good that it does not differ from the rest of the felt.*

(T4, 00:53:48): ... *What it [the dry content of paper] depends also significantly on is the felt type… especially in the extended nip press. In the beginning there were completely wrong types of felts. When the right types of felts were found, the dry*

content rose significantly and the running speed could be increased… The extended nip was an invention which caused many changes in the press section…

Explanation: These quotations represent good examples of the series of consequences to the mental representations of the design engineers caused by the changes in the increased running speed of the paper machines and the emergence of the extended nip press. A thicker felt was needed. The felt changing procedures were necessary to be changed. At first the whole construction of the press section was redesigned to a cantilevered model, where the stiff felt could be changed. This was a rather expensive construction, but was an established solution for over two decades. However, modern technology allows for the use of joints in the thick felt. The felt can be thread into the press and then the ends are joined, without the expensive cantilever constructions.

EXAMPLE 10: Figure 7 is a scheme of the total assemblage of the belt roll of the ENP-press. It illustrates the fine-grained diversity of the variables, which have been under the thought processes of the participating engineers during the design process according to our empirical material.

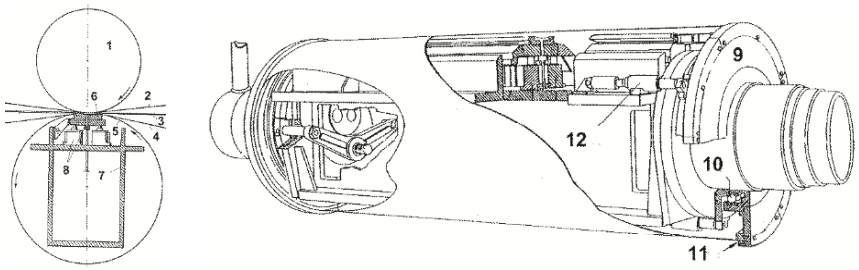


Figure 7. Schematic illustration of the total assemblage of an ENP roll. Adapted from patent publications (Finland 2000) and (PCT 1993b).

The components of the resulting overall construction and the respective central (technical) design attributes are listed in Table 1. It contains a condensed summation of the attributes of design thinking, which were discovered from our empirical material. Each of the problems had to be solved and incorporated into the "final" solution (see Nevala, in press, for a more detailed discussion).

Furthermore, in order to make this assembly a functioning part of larger machinery many other details must have been solved (not included in Figure 7 or Table 1). There are also equipments for power supply, hydraulics units, the means for applying the pressing force, water removal equipment, the support structures and frames of the press unit, felt changing means, mantle

changing means, etc. All of these constructs and arrangements still include a bunch of fine-grained intricacies.

TABLE 1. Examples of the technical design attributes of an ENP unit. The numbers of the table refer to Figure 7.

Component	Central technical attributes
1. Counter roll	- Surface properties - Diameter - Length - Deflection compensation - High load carrying capacity
2. and 4. Dewatering felts	- Thickness - Porosity - Fiber qualities - Tension - Initial moisture - Water removal properties - Stability of properties over time
3. Paper web + water	- Requirements for the paper grade - Dewatering process
5. Flexible mantle	- Absolute impermeability - Surface properties - Sliding properties - Flexibility - Resilience - Dimension and form stability - Heat resistance - Endurance - Void volume in the mantle surface to improve water removal
6. Press shoe	- Surface geometry - Surface properties - Lubrication
7. Supporting beam inside the Roll	- High load carrying capacity - Space and placing for cylinders and hydraulic oil feeding and removal equipment
8. Hydraulics	- Loading cylinders and lubrication oil feeding in the shoe - Hydraulic center (not shown in the figure): Loading hydraulics, lubrication hydraulics, oil cleaning and cooling systems
9. Roll end plates	- Minimum weight (The flexible roll is rotated by the counter roll through friction forces, i.e. the flexible mantle drives the end plates)
10. Bearings	- Minimum friction (see above)
11. Mantle-end plate fastening	- Enables the flexible mantle assured fastening without oil leakage - Enables the flexible mantle to be fastened straight
12. Mantle stretching devices	- Enables the flexible mantle to run straight and round

The content structure of the constructive mode of thinking is evidently different from apperception, restructuring and reflective thinking. In constructive thinking, the elements of the final design are organized into a self-consistent form. The resolved sub-problems find their places in the plan. This means that the content of the constructive representation is consistent as in apperceiving, but simultaneously it also contains a large set of supplementary elements.

These examples only represent a very small part of the total set of solved problems. Nevertheless, it gives an idea to what sort of things must be bound together in constructive thinking.

Discussion

Typically, in the constructive mode the hypothetical but already working solutions find their place in the whole in what the engineers are building. In principle, constructive mode gives the "eventual"[2] form to the design solutions. It forms the realizations of the design ideas and makes the realization of design ideas possible to begin. When something has got the form that can be incorporated into the final plan, it has its form and place in the whole. It need not be changed but in rare cases we can think that the problem has been solved in this respect.

Construction differs essentially from restructuring and reflection, because it produces integrated representations. It does not focus on inconsistencies. It differs from apperception because of its focus. When apperception entails one sub-problem and a suitable solution to it, construction integrates large groups of solutions together. For this reason it is normally impossible to keep all the elements in mind. It is necessary to move the focus between the elements to find a place for all of them. In this way, construction exceeds the limits of apperceptive representations.

5. General Discussion

We have addressed two different questions here. Firstly, we have considered the sub-processes of design thinking from the contents point of view. Secondly, we have focused on developing the analysis of design thinking from the contents point of view. To the very end we shall briefly discuss about the modes or phases of design thinking and secondly the relations of computational and content-based analysis of human thinking.

---

[2] Quotes are used around "final" and "ultimate", because such attributes can be given to engineering solution only in relative sense. Improvements are on the way all the time.

We have suggested a model with four different modes. Apperception means activation of one pre-dominant mental representation. Restructuring is a qualitatively different mode from apperception. It is the process of shifting one representation to another. The reflective process of thinking means that the new hypotheses generated by restructuring processes will be assessed. It is possible to see in the interviews and documents how design engineers' minds have alternated between various possibilities. They have looked for the optimal way of solving the problem. They have to choose between alternative problem and solution representations. They have to compare them and make choices. Finally, one must incorporate the selected solution to the whole. One must decide how the solution is connected to the whole. The acceptance makes the constructive mode different from others. The term mode is selected, because differently from phases, we do not suppose linear ordering of the sub-processes. The presented system is analogous with the stages found in analyzing a chess players' thinking (Saariluoma 1995). This suggests that the modes are not only specific to paper machine design.

The abstraction of modes is naturally a small set of content-based design analysis. Nevertheless, it allows us to illustrate the possibilities of this new approach. To make things clearer it is good to compare the content-based approach with one of its major alternative. This is capacity-based analysis of design thinking. It is well known that human limited capacity explains thought errors also in design (Anderson, Farrell and Sauers 1984; Johnson-Laird 1983; Kavakli and Gero 2003; Miller 1956; Simon 1974). As we can fill the capacity with any contents, this approach cannot explain content-based phenomena in design thinking. From the capacity point of view, it is equivalent whether memory is filled with paper or aircraft engineering knowledge, but from contents point of view, the difference is essential.

One may naturally be skeptical and ask why we are interested in modes. One answer is practical. We work to reconstruct a decades long design processes. The information we receive from one interview is a very small part of the final whole. We need to have a very clear idea about the dynamics of the thought processes so that we are able to direct our reconstruction process. Many hypotheses live for seconds but some may live for years and eventually become discarded and restructured. This means that we need an overall schema of human design thinking, which can be used to organize the collection of knowledge, the collected knowledge, select essential parts of the data and to separate non-process aspects of thought contents from the process itself.

References

Akin, O: 1980, *The Psychology of Architectural Design*, Pion, London.

Altshuller, G: 1999, *The Innovation Algorithm. TRIZ, systematic innovation and technical creativity,* Transl. Lev Shulyak, Technical Innovation Center, Worcester, MA.

Anderson, JR, Farrell, R amd Sauers, R: 1984, Learning to program lisp, *Cognitive Science* **8**: 87-129.

Bartlett, F C: 1958, *Thinking,* Allen and Unwin, London.

Cross, N, Christiaans, H and Dorst, K: 1996, *Analyzing Design Activity*, John Wiley, New York.

Dewey, J: 1910, *How We Think?* McMillan, New York.

Dey, I: 1993, *Qualitative Data Analysis: A User-Friendly Guide for Social Scientists*, Routledge, London.

Denzin, NK and Lincoln, YS: 1994, Introduction: Entering the field of qualitative research, *in* K Denzin and YS Lincoln (eds), *Strategies of Qualitative Inquiry*, Sage London, pp. 1-17.

Duncker, K: 1945, *On Problem Solving, Psychological Monographs*, Volume 270, APA Washington.

Dym, CL: 1994, *Engineering Design. A Synthesis of Views*, Cambridge University Press, Cambridge.

Ehrlenspiel, K: 2003, On the importance of the unconscious and the cognitive economy in design, *in* U Lindemann (ed), *Human Behavior in Design. Individuals, Teams, Tools*, Springer, Berlin, pp. 25-41.

Ericsson, KA and Simon, HA: 1980, Verbal reports as data. *Psychological Review* **87**: 215-251.

Ericsson, K and Simon, H: 1984, *Protocol Analysis*, MIT-press, Cambridge, Mass.

de Groot, AD: 1965, *Thought and Choice in Chess*, Mouton, The Hague.

Gullichsen J and Paulapuro H (eds): 1998 – 2000, *Papermaking Science and Technology*, Books 1- 8, Fapet, Helsinki.

Hodder, I: 1994, The interpretation of documents and material culture, *in* NK Denzin and YS Lincoln (eds), *Handbook of Qualitative Research*, Sage, London, pp. 703-715.

Johnson-Laird, PN: 1980, Mental models in cognitive science, *Cognitive Science* **4**: 71-115.

Johnson-Laird, PN: 1983, *Mental Models. Towards Cognitive Science of Language, Inference, and Consciousness,* Cambridge University Press, Cambridge.

Justus, EJ and Cronin, DC: 1982, Development of extended nip press, *TAPPI 1982 Papermakers Conference,* Atlanta, GA, pp. 39-44.

Kant, I: 1781, *Kritik der Reinen Vernunft*. [The Critique of Pure Reason] Philip Reclam, Stuttgart.

Kavakli, M and Gero, JS: 2003, Strategic knowledge differences between an expert and a novice designer, *in* U. Lindemann (ed), *Human Behavior in Design. Individuals, Teams, Tools*, Springer, Berlin, pp. 42-52.

Köhler, W: 1930, *Gestalt Psychology*, London.

Köhler, W: 1956, *The Mentality of Apes*, Penguin Books, Harmonsdworth.

Leibniz, GW: 1704, *New Essays on Human Understanding*, Cambridge University Press, Cambridge.

Lindemann, U: 2005, *Methodische Entwicklung Technischer Produkte*. Berlin, Springer.

Miller, GA: 1956 The magical number seven, plus or minus two: Some limits on our capacity for processing information, *Psychology Review* **63**: 81-97.

Nevala, K: 2005a, Mechanical engineering way of thinking in a large organization, a case study in paper machine industry, *AEDS Workshop*, Pilsen.

Nevala, K: 2005b, *Content-based Design Engineering Thinking. In Search for Approach*. Jyväskylä Studies in Computing 60, Jyväskylä University Printing House, Jyväskylä.

Newell, A and Simon, HA: 1972, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ.

Norman, E, Cubbitt, J, Urry, S and Whitaker, M: 2000, *Advanced Design and Technology*, Pearson, Edinburgh.

Pahl, G, Beitz, W, Feldhusen, J, Grote, KH: 2003, *Konstruktionslehre. Grundlagen Erfolgreicher Produktentwicklung. Methoden und Anwendung,* Springer, Berlin.

Saariluoma, P: 1984. *Coding Problem Spaces in Chess*, Sociedtas Scientiarum Fennica. Helsinki.

Saariluoma, P: 1990, Apperception and restructuring in chess players' problem solving, *in* KJ Gilhooly, MTG Keane, RH Logie and G Erdos (eds), *Lines of thought: Reflections on the Psychology of Thinking*, Wiley, London, pp. 41-57.

Saariluoma, P: 1992, Error in chess: Apperception restructuring view, *Psychological Research* **54**: 17-26.

Saariluoma, P: 1995, *Chess Players' Thinking*, Routledge, London.

Saariluoma, P: 1997, *Foundational Analysis: Presuppositions in Experimental Psychology*, Routledge, London.

Saariluoma, P: 1998, Adversary problem solving and working memory, *in* R Logie and K Gilhooly (eds), *Working Memory and Thinking*, Erlbaum, Hove, pp. 115-138.

Saariluoma, P: 2001, Chess and content-oriented psychology of thinking, *Psicologica* **22**: 143-164.

Saariluoma, P: 2002, *Ajattelu Työelämässä* [Thinking in work life], WSOY, Helsinki.

Saariluoma, P: 2003, Apperception, content-based psychology and design, *in* U Lindemann (ed), *Human Behavior in Design. Individuals, Teams, Tools*, Springer, Berlin, pp.72-78.

Saariluoma, P. and Hohlfelt, M: 1994, Apperception in chess players' long-range planning, *European Journal of Cognitive Psychology* **6**: 1-22.

Saariluoma, P, and Maarttola I: 2001, Spatial mental content and visual design, *in* JS Gero, B Tversky and T Purcell (eds) *Visual and Spatial Reasoning in Design II,* Key Centre of Design Computing and Cognition, University of Sydney, pp. 253-266.

Saariluoma, P and Maarttola, I: 2003, Stumbling blocks in novice building design, *Journal of Architectural and Planning Research* **20**: 244-254.

Saariluoma P, Nevala, K and Karvinen M: (2005a) Content-based design analysis, *in* JS Gero and N Bonnardel (eds) *Studying Designers '05,* Key Center of Design Computing and Cognition, University of Sydney, pp. 213-228.

Saariluoma P, Nevala K and Karvinen M (2005b) The modes of design engineering thinking, *in* JS Gero and ML Maher (eds)*, Computational and Cognitive Models of Creative Design VI*, University of Sydney, pp. 10-14.

Shigley, JE and Mitchell LD: 1990*, Mechanical Engineering Design*, McGraw, New York.

Simon, HA: 1969, *The Sciences of the Artificial*, MIT Press, Cambridge, MA.

Simon, HA: 1974, How big is a chunk? *Science* **183**: 482-488.

Stout, GF: 1896, *Analytic Psychology*, Macmillan, New York.

Suh NP: 1990, *The Principles of Design*, Oxford University Press, New York.

Turing, A: 1936, On computable numbers, with an application to the Entscheidungs problem, *Proceedings of the London Mathematical Society,* **42**: 230-265.

Turing, A: 1950, Computing machinery and intelligence, *Mind* **49**: 433-460.

Tversky, B: 2003, Sketches for design and design of sketches, *in* U Lindemann (ed), *Human Behavior in Design. Individuals, Teams, Tools*, Springer, Berlin, pp. 25-41.

Wallas, G: 1926, *The Art of Thought*, Hartcourt, New York.

Watson, JB: 1914, *Behaviourism*, Kegan Paul, Trench, Trubner, London.

Wertheimer, M: 1945*, Productive Thinking*, Harper and Brothers Publishers, London.

Vincenti, WG: 1990, *What Engineers Know and How They Know It. Analytical Studies from Aeronautical History*, The Johns Hopkins University Press, Baltimore.

Visser, W: 2003, Dynamic aspects of individual design activities. A cognitive ergonomics viewpoint, *in* U Lindemann (ed), *Human Behavior in Design. Individuals, Teams, Tools*, Springer, Berlin, pp. 87 -96.

Wittgenstein, L: 1953*, Philosophical Investigations*, MacMillan. New York.

Wittgenstein, L: 1967*, Zettel*, Basil Blackwell, Oxford.

Wittgenstein, L: 1980, *Remarks on the Philosophy of Psychology*, Basil Blackwell, Oxford.

Wundt, W: 1880*, Logik, I* [Logic I], Ferdinand Enke, Stuttgart.

Patent publications:

Canada 1948, Process of an Apparatus for Extracting Liquid From Pervious Sheet Material, Patent No. 452,200, October 26, 1948.

Canada 1975, Hydrodynamically Loaded Web Press, Patent No. 969398, June 17, 1975.

Finland 1995, Laakeripesien välinen kytkentärakenne pitkänippipuristimella [Coupling Construction between Bearing Housings in an Extended Nip Press], Finnish Patent Application No. 963702, August 3, 1995.

Finland 2000, Press shoe for a long-nip press, Finnish Patent Application No. 20002412, November 2, 2000.

Germany 1972, Nasspresse zum Entwässern einer Faserstoffbahn, Offenlegungsschrift 2108423, 7. September 1972. (Anmelder: VEB Papiermachinenwerke Freiberg; X 9200 Freiberg, DDR.)

Germany 1985, Presswaltze, Offenlegungsschrift 3338487, 2. Mai1985.

PCT 1991, Press Roll, International Application Number: PCT/SE91/00325.

PCT 1993a, A Shoe Type Press, International Application Number: PCT/SE93/00390.

PCT 1993b, Press Shoe, International Application Number: PCT/SE92/00874.

USA 1966, Machine for Making Extensible Paper, Patent No. 3,269,893, August 30, 1966.

USA 1974, Anti-rewet Membrane for an Extended Press Nip, Patent No. 3,840,429, October 8, 1974.

BUILDINGS AND AFFORDANCES

ALEXANDER KOUTAMANIS
Delft University of Technology, The Netherlands

Abstract. The notion of affordances has been used to represent functionality and usability in several design areas. The paper considers its applicability to architecture and buildings. It discusses a distinction between the affordances of building elements and spaces, and a number of dimensions for the mapping of different aspects.

1. Affordances

The term 'affordance' has been coined by the psychologist James Gibson to indicate the actionable properties the environment offers to an animal (Gibson 1977; 1979). According to Gibson perception does not aim at an internal representation of the visual world but at the detection of such relationships between the environment and the animal. Affordances exist in the environment and can be linked to its physical properties but have to be measured relative to a particular animal. For instance, an approximately horizontal and flat surface that is sufficiently large and rigid for a particular animal affords support to the animal. Nevertheless, the affordances of an environment are independent of the individual animal's ability to perceive them and do not change when the individual's needs and goals change. A transparent horizontal surface may afford support to an infant, even though the infant is reluctant to crawl over such a visual cliff (Gibson and Walk 1960). Gibson claims that affordances are independent of the individual's experience and culture, but in many cases action and interaction arguably presuppose prior experience with a similar environment. Grasping an object, for example, can be generalized from early experiences in infancy to a large number of environments, while writing on an object probably relates to more specific experiences with the same media (writing implements and writing surfaces). Gibson names this type of knowledge *mediated* or *indirect* knowledge, i.e. second-hand knowledge with a strong cultural dimension.

Gibsonian affordances are an attractive notion, primarily because of their immediacy. However, Gibson provides few examples, mostly obvious stereotypes that illustrated his main points. The resulting vagueness of the

J.S. Gero (ed.), Design Computing and Cognition '06, 345–364.
© 2006 *Springer. Printed in the Netherlands.*

term and its application did little to promote research into the subject until the publication of *POET: The Psychology of Everyday Things* (Norman 1988), republished as *The Design of Everyday Things* (Norman 2002). Norman deviates from Gibson's use of affordances by considering them in relation to both actual and perceived properties which determine how an object could be used. In POET perception by an individual, with all its personal and cultural bias, is a determinant of affordances. The difference between the two definitions becomes evident when we consider the example of a hidden door in a paneled room. For Gibson the hidden door affords passage, while in POET it is seen as a case of a forcing function, i.e. an attempt to reduce the usability of the door in order to achieve another goal. Gibson relates affordances to the action capabilities of the animal, while Norman stresses the mental and perceptual capabilities of the actor (perceived affordances). In POET affordances depend on culture and past experience, i.e. learning through social interaction and experimentation.

Another departure from Gibson is that POET concentrates on man-made objects and relationships between design and use. Affordances provide strong clues to the operations of objects and suggest the range of possibilities for use. Norman's starting point is the apt observation that many people experience trouble with common everyday tasks such as opening a door or turning on a light, while at the same time they prove capable of mastering complex technologies and challenges like computer programming. He proposes that this is due to faulty design rather than the ineptitude of the users, as much of our everyday knowledge resides in the world and not in the head (which is a main argument of Gibson's approach to visual perception). The availability of knowledge in the world means that precision in behavior is not impeded by imprecision of knowledge in the head (combination of declarative and procedural knowledge). POET argues that when designers take advantage of affordances, the user knows what to do just by looking. Although complex objects or situations may require supporting information, simple tasks should not – otherwise the design has failed. Good use of affordances in the design of an object can help reduce the level of cognition and learning time required to use it. This should also be the case in architecture and building: most uses of the built environment should not require any additional information.

POET has been influential in various design disciplines, such as product design and human-computer interaction. There was, however, ambiguity in Norman's original definition and use of affordances that resulted in widely varying uses of the concept, even as a synonym of "advantage" or "property". The main cause of confusion seems to have been that POET "collapsed two very important but different, and perhaps even independent, aspects of design: designing the utility of an object and designing the way in which that utility is conveyed to the user of the object. Because Norman has

stressed (but not entirely limited himself to) perceived affordances, he has actually favored the latter of the two" (McGrenere and Ho 2000). Such misunderstandings have stimulated corrective interventions by Norman who made efforts to clarify that POET focuses on *perceived* affordances because "the designer cares more about what actions the user perceives to be possible than what is true" (Norman 1999). He emphasized the distinctions between the user's conceptual model, physical constraints, (cultural) conventions and differences between perceived and real affordances.

A review of recent research literature suggests that the discussion focuses more on the notion of affordances (using superficial examples) than on thorough analyses of its applicability in different areas. Still, the relevance of affordances to a good design seems to have become an established concept in several design disciplines, despite a number of problems that remain to be solved satisfactorily. These include:

1. Differences in affordances between designers and users or between different types and classes of users (both physically and culturally, e.g. between children, adults and the elderly or between European and Japanese users of a chair).
2. The relationship of such differences with the difference between perceived and real affordances.
3. Ambiguity towards design innovation: POET and subsequent studies of affordances in design tend to overestimate the significance conventional concepts and constraints in an attempt to satisfy apparent user requirements ('natural' designs).
4. Uncertainty concerning the form of design guidance: approaches based on affordances may have proscriptive undertones leading to stereotypical or deterministic designs, while affordances seem to promote a more fundamental analysis of usability and functionality.

Despite such problems, affordances are an interesting notion also for architectural design. In a correlation of affordances and building design Tweed stresses the holistic character of affordances and their potential in integrating different functionalities, including aesthetics (Tweed 2001). Affordance theories suggest that human interaction with the built environment is largely conditioned by the affordances of building elements and spaces. These should allow for direct recognition of possibilities in any setting, efficient fuzzy planning of actions, and a 'natural' manipulation of building elements and spaces. The similarities between these consequences and the casual or cavalier attitude of many designer and users of the building environment with respect to functionality and usability are striking. A frequent objection to analytical tools for supporting design by e.g. explicitly structuring and analyzing a brief or stating detailed accessibility criteria is based on the assertion that the capable architect caters for such aspects

intuitively. Equally intuitive and direct are the ways most users approach and manipulate buildings: it appears that they take quite a lot for granted and that their expectations are usually met by the building. Buildings should not require extensive and detailed explanation of how they work (e.g. a user manual) but be immediately evident on the basis of direct, meaningful relationships with the users' expectations (even though travelers may be puzzled by foreign types of fixtures). Most problems in the use of buildings are not due to cultural and individual differences but are caused by design limitations (e.g. the size or shape of a space) or incompatible use specification (e.g. large furniture in a small space). Affordances promise integration of different viewpoints (architects, engineers, clients, users) and continuity, i.e. compatible expressions of functionality and usability throughout the life cycle of a building (briefing, design and use). This holds promise for the codification of design knowledge: affordances could support direct matching of an existing building or type to a specific brief, thus allowing for early evaluation and refinement of design or briefing choices.

2. Building Elements

It is interesting and rather amusing that doors, a basic class of building elements, one of the favorite examples in illustrations of affordances. In POET Norman stresses the simplicity of door functions (one either opens or shuts it) and proceeds to illustrate how designers can eliminate natural signals that reduce the visibility of affordances by allowing aesthetics to get in the way of understanding how to interact with a door (not knowing whether to pull, push, slide etc.). The evaluation of door affordances usually focuses on door handles and their relationships with the way users can open and close a door. The evaluation is based on:

1. The mapping of human anatomy on the form and operation of the door handle: a lever and a pull and push bar are held in a similar manner but in a different orientation; a knob and a lever are held differently but can both turn in order to release the latch, Figure 1.
2. The physical constraints that constrain the mapping: the size of a handle indicates how many fingers or hands could be used to hold and apply the appropriate force to it.

The combination of the two should determine the way a user operates the door: a lever or knob invites the user to turn it and then pull or push, a pull and push bar indicates that one should either pull or push, and a metal plate only affords pushing, Figure 2. Other combinations would confuse the user and should therefore be avoided. This example makes clear that affordance studies tend to focus on design as communication and attempt to promote the integration of visual clues in a framework for perception and action. They

realize that the information specifying an affordance is not the same as the affordance itself but at the same time they can be too selective (by focusing on just part of the information) and rather deterministic: in the example of Figure 2, the lever handle actually affords all four possible actions (turn, pull, push, slide), the pull and push bar three (no turn) and the plate one (push). This suggests that an appropriately shaped lever or a pull and push bar that also releases the latch could be used for all types of doors. Such combinations are frequently encouraged in architecture (and product design).

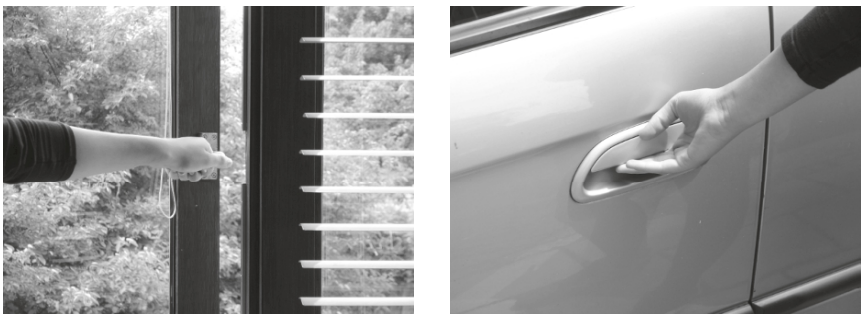


Figure 1. Mapping of hands on different door handles.

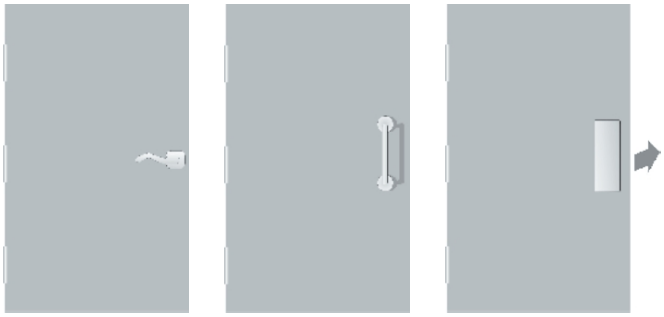


Figure 2. Affordances of door handles: lever, pull and push, plate (source: www.infovis.net).

The affordances of building elements such as doors and windows have a similar scale and user interaction to the majority of objects discussed in affordance studies. However, architectural design generally involves a wider functional scope and greater flexibility requirements. We can distinguish between two levels of functional abstraction (Tweed 2001):

1. *Spatial level*: a door affords communication between two spaces, as well as separation between two spaces optically, acoustically etc.
2. *Interaction* with the door itself in order to achieve this communication or separation.

The spatial level is important for the formulation of use expectations and goals, as well as for the recognition of visual clues pertaining to affordances. The former is arguably a main point of convergence for designers and users: the design of a building should also generate consistent affordances that improve functionality and usability. Spatial aspects should inform users in a direct and non-trivial manner about the intentions of the architect and the behavior of the design. Figure 3 is a popular illustration of a misaffordance: by designing both fixed and opening parts of the opening in the same way, the door is inadequately indicated and the user has no idea where to go (Evans and Mitchell McCoy 1998). However, if the approach to the door is clearly indicated by e.g. the paving, users experience little uncertainty in moving towards and through the door, despite its vague design.

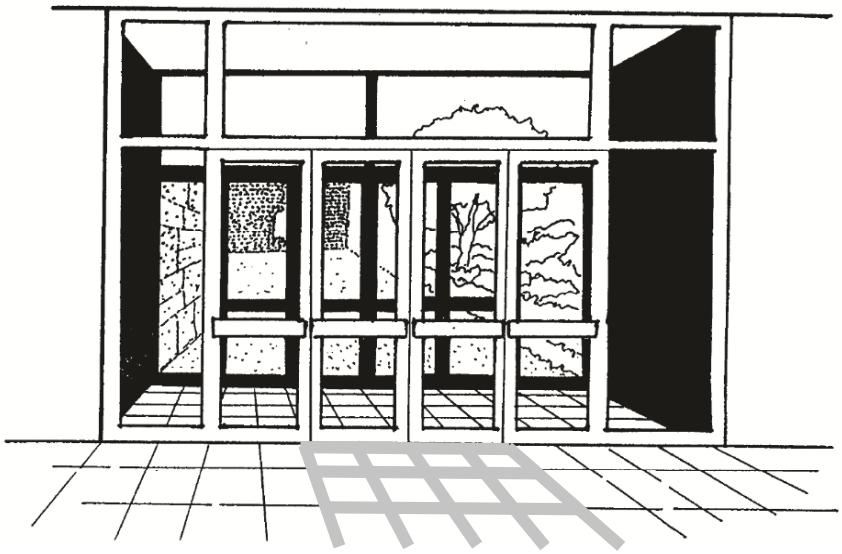


Figure 3. Contextual clues: the approach to the door as a correction of misaffordance –adapted from (Evans and Mitchell McCoy 1998).

Recognition of relevant visual clues involves not merely the door handle but also other critical features of a door, e.g. the visibility of hinges (which are strangely ignored in affordance studies). These may indicate the type of the door with more accuracy than the handle, as well as additional characteristics such as the swing of the door. From a spatial viewpoint the position of the door in the wall is probably more interesting. Of the two most popular types, an inwards opening hinged door is usually placed on the same plane as the interior surface of the wall, Figure 4, while an outwards opening hinged door is recessed, Figure 5. In the outwards opening case this results

into a cavity that is readily perceived and a known clue. The origin of the cavity most probably lies in construction, but one cannot ignore the association of the small cavity with the bigger hole behind it (the space).

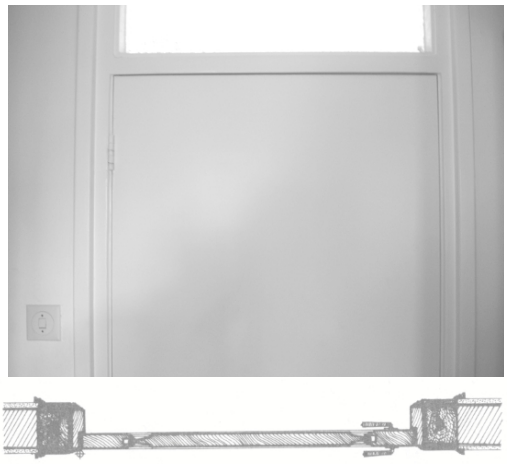


Figure 4. Inwards opening door.

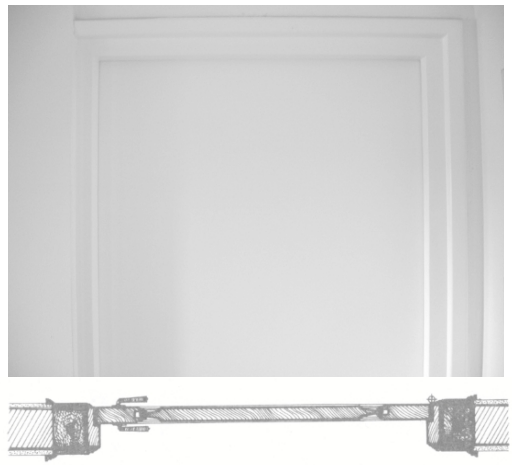


Figure 5. Outwards opening door.

Interaction with the door remains based on the mapping of the users' anatomy and actions onto critical features, including the interfaces with the user (e.g. door handle). Mapping involves several interrelated dimensions:

1. *Physical/mechanical:* this dimension refers to the constraints that determine the way an object can respond directly to the actions of a user, e.g. size considerations or the matching of degrees of freedom

between the user's hand and the door handle, Figure 6. These limit the relationship of an object to other objects in specific ways.

2. *Perceptual:* purely formal features that indicate general preferences and possibilities, e.g. that the extremities of an object usually afford handling, Figure 7. It is important that the identification of such features relies on universal principles such as transversality or colinearity (Hoffman and Richards 1985; Kim et al. 1987). It is not accidental that the user interface of a door is normally a small protruding subpart, i.e. something that can be readily recognized against a background of flat panels, Figure 8.
3. *Semantic* constraints refer to the interpretation of an object on the basis of expectations that may have a physical, perceptual or cultural background. Visually and mechanically a lever-type handle suggests two possibilities for mapping a hand but also a clear preference order, Figure 9. This order is reversed in the case of a door knob. Semantic constraints also underlie the identification and repair of missing or misaligned parts, e.g. a door handle that has fallen off.

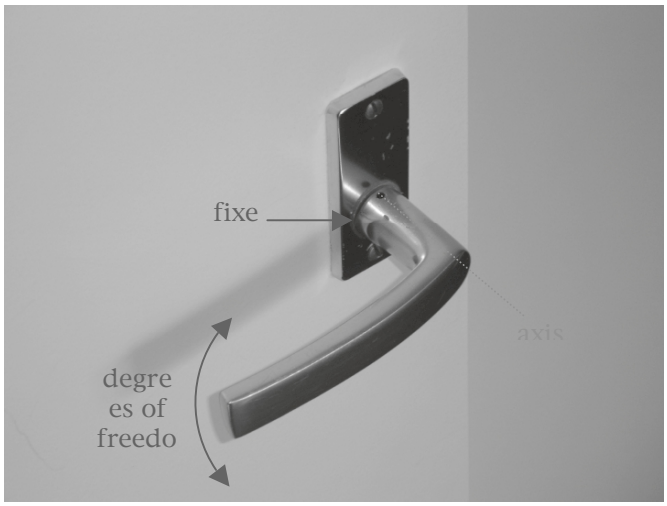


Figure 6. Physical / mechanical constraints.

4. *Cultural:* the relationship between affordances and cultural constraints remains troublesome, even when we account for the influence of design (Ingold 1992). There are, however, constraints that can only be called cultural, e.g. that a red sign by a door indicates an emergency exit, the strong preferences for canonical views, the expectation that text signs are the right way up and that arrows indicate the direction one should follow in order to reach the indicated place. The roles of cultural constraints and especially

custom may not be always apparent in a slow, old area such as architecture – at least not in the spectacular way other areas are experiencing frequently arbitrary changes (e.g. the form of thumb keyboards for text messaging), which derive more from the adaptability of the user than good design. Still, there are some clear examples of cultural influences in building affordances, e.g. the expectation that most doors in an air terminal open automatically as the user approaches in relation to the absence of user interfaces on these doors. Few adults experience discomfort with such doors, unless of course the automatic doors fail to match their speed.

Figure 7. Perceptual constraints: extremities are for grasping.

Figure 8. The user interface of a door is recognizable as a small protruding part.

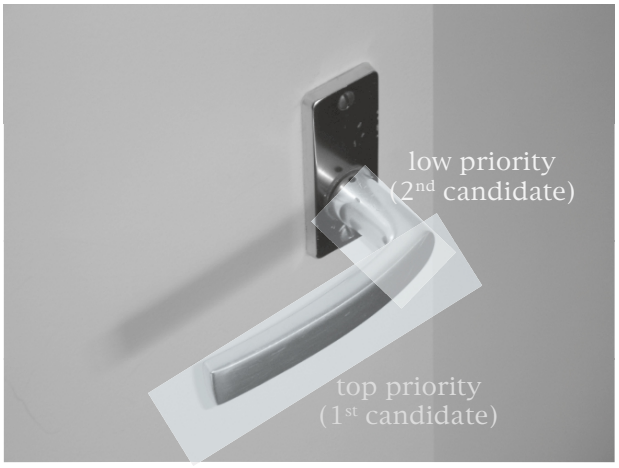


Figure 9. Semantic constraints.

Any analysis of affordances in building elements should not fail to identify the distorting adaptability of users and the resulting increase of flexibility. The ways we treat building elements may deviate from their intended uses but remain nevertheless well within what we would consider as 'normal' behavior, Figure 10 and 11. In most cases the functionality and usability of building elements contain substantial gray areas that are completely unrelated to misaffordances. Users are notorious for effortlessly recognizing and exploiting the affordances hidden in such gray areas, even though this might conflict with the designer's intentions and norms.

3. Spaces

Spaces deviate from the common examples and subjects of affordance studies. They offer few tangible forms that permit the mapping of individual human functions. Moreover, they generally lack the handy interfaces that allow interaction with 'solid' objects. Even worse, such interfaces tend to adopt a naive view of space and architecture. For instance, POET praises the 'natural' mapping of an entity onto relevant controls (e.g. light switches on a scaled floor plan instead of an array). One could claim that a higher degree of abstraction is necessary for dealing with the complexity that is caused by the flexibility and adaptability of space in relation to user activities.

The two main levels of abstraction proposed for building elements also apply to the functional patterns that are accommodated in a space:

1. The *spatial* level refers primarily to the internal structure of these patterns and includes their basic relationships with the environment, i.e. relationships with the basic surfaces of a space like the floor.

2. The *interaction* level concerns the mapping of these patterns to the spaces that accommodate them in a correlation of form and function.

Figure 10. Sitting affordances of a bench.

Figure 11. Sitting affordances of a fence.

The mapping dimensions proposed for building elements (physical/ mechanical, perceptual, semantic and cultural) also apply to spaces and have the same characteristics. The main problem is which information should be mapped onto spaces and how. Reducing space to the surfaces of building elements that bound them is a minimal option that allows definitions such as that a floor affords walking, standing and placing furniture on it or that a wall affords leaning against it and hanging pictures on it. However, this returns a rather incoherent network of loosely connected basic affordances that does little justice to the spatial thinking of both designers and users.

Adding users as independent entities in a design representation offers a less deterministic alternative to activity modeling. This can be achieved by means of user interfaces that permit e.g. walkthroughs in a virtual environment and allow different users to experience the affordances of a

design. Similar results can be achieved with virtual users, e.g. analysis of a design on basis of user representations to identify areas accessible to a user type or occupiable by an activity (Koutamanis et al. 2001; Tweed 2001). Such techniques can make affordances explicit but mostly in a procedural manner that aggregates user experiences and local analyses. This arguably weakens the immediacy of affordance recognition and utility.

An alternative to such representations and analysis can be derived from conventional architectural knowledge and technology. The orderly collection of verifiable information on use patterns and their functional requirements has been one of the priorities in both architectural research and practice. The results have formed the basis for professional handbooks such as Neufert's *Bauentwurfslehre*, Figure 12 and drafting templates, Figure 13. These are more than indications of sizes for various objects or handy drawing aids. They also incorporate information on relationships between objects and spatial arrangements based on explicit use constraints. For instance, they indicate how many chairs can be placed around a table of a given size and form on the basis of the space required for the affordances of e.g. sitting in a chair at a dining table. Architects use such information as reference for the design and analysis of functionally intricate situations that require precision in behavior and unambiguous recognition of affordances, Figure 14. It provides an insightful and operational correlation of form and function which is further enhanced by the mental aggregates designers form through the integration of multiple patterns and constraints (Koutamanis 1997).

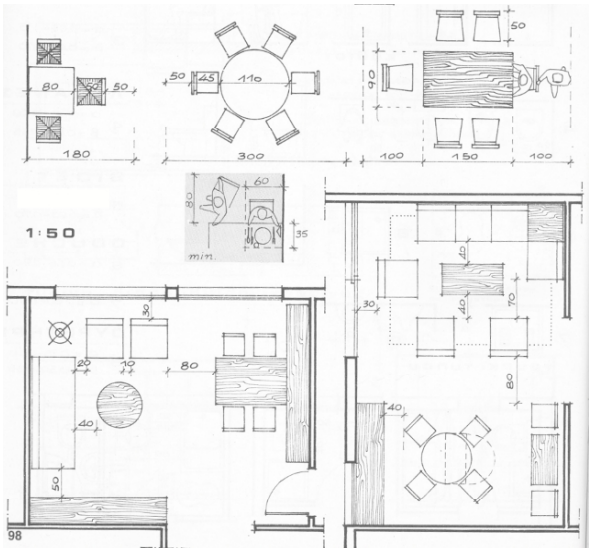


Figure 12. Spatial arrangement examples from an architectural handbook.

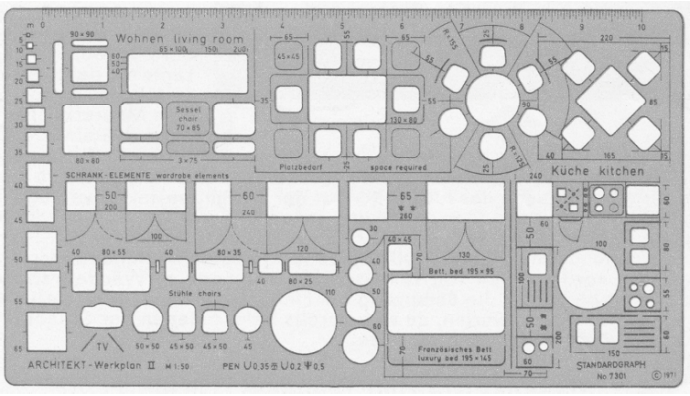


Figure 13. Architectural drafting template (Standardgraph).

By mapping such patterns and their constraints onto the form of a design we can recognize affordances at a number of abstraction levels that permit quick, transparent changes of focus from e.g. a single user's interaction with a space to a group of users and their interaction with the built environment and each other. This multi-level abstraction and the flexibility of choice implied by the underlying functional patterns are essential for the correlation of designers' and users' perception of affordances. The feeling of helplessness users experience with misaffordances in the built environment is often the unnecessary consequence of insufficient understanding of spatial aspects, leading to e.g. doors bumping into each other and other problems that a designer should resolve by default. Architects can also be insensitive to practical problems that conflict with higher, usually aesthetic norms. In both cases it is important that correlation of designers' and users' perceptions also promotes design innovation or at least reduces the danger of falling back to stereotypical solutions and arrangements.

The correlation of functional patterns and form is frequently based on transformations and affordances that require professional design knowledge and experience but many aspects of a design solution also refer to a general understanding of space. On the basis of universal principles such as transversality and colinearity both design professionals and lay users are able to segment the built environment into more or less the same components and arrive at an objective description that underlies many semantically or culturally enhanced interpretations (Biederman 1987; Hoffman and Richards 1985; Kim et al. 1987). The affordances of many of these components are common to both designers and users and can add to the constraints of a design. For instance, an alcove generally invites activities characterized by a higher degree of privacy. Placing a bed or a solitary armchair in an alcove is therefore a more or less standard reaction that may influence the overall arrangement of activities in a building.

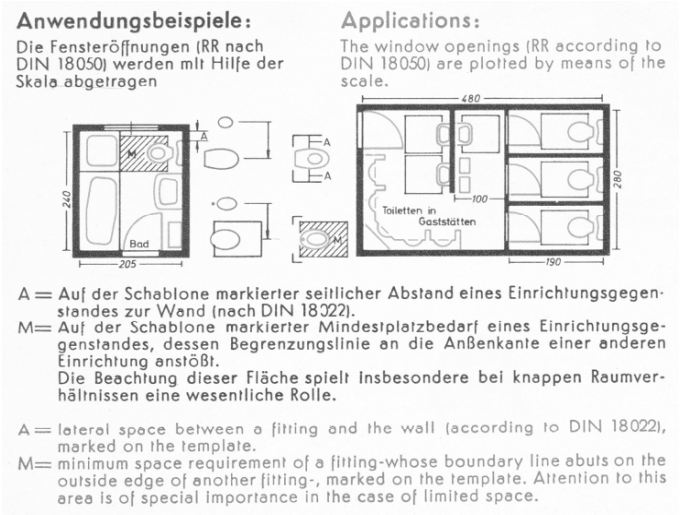


Figure 14. Application instructions for a drafting template (Standardgraph).

A primary aspect of affordances is their static character, even though they refer to dynamic activities and situations. This can be seen as a form of informational economy that agrees with the idea of information being available in the world and permits higher abstraction and efficient processing without loss of specificity. Still, mapping dynamic use patterns especially onto spaces probably has significance for building design and in particular for the adjustment of goals and actions on the basis of direct feedback from user-building interaction. Making explicit a sequence of actions in these dynamic patterns makes possible the verification and refinement of expectations concerning functionality and usability. For example, the constraints indicated in Figure 14 are generally sufficient for the mapping of critical points such as opening the door of a WC cubicle but may obscure difficulties like having to walk sideways after closing the door so as not to collide with the walls of a small cubicle or what happens two persons have to move in the same bathroom. Designers can be selective in what they consider to be critical and rather negligent of what they deem to be less important simply because users can be flexible, adaptable and tolerant to design limitations despite constant irritation and frustration. However, reactions to such selectivity should not lead to overestimation of the influence of architecture. The built environment is generally background to human goals and actions – rarely the subject itself.

5. Implementation

In an experimental implementation of affordances that explored the inclusion of the notion in a design representation and the connection

between affordances in briefing and designing, the mapping of the spatial dimension was based on the mechanism of *local coordinating devices* (Koutamanis 1997). This was developed for the representation of local constraints into autonomous entities focused on (configurations of) critical architectural elements – as opposed to turning the constraints into properties of these elements. Local coordinating devices allow for a higher degree of abstraction and generalization than plain constraint networks because they express requirements on classes of entities and related activities.

In the design representation the implementation investigated the differences between affordances of building elements and of spaces, i.e. the actionable properties of critical building elements and the accommodation potential of spaces. The affordances of building elements derived primarily from the same functional and structural constraints that define a local coordinating device but also related to the perceived affordances of an element in a specific context, for example the visibility of an entrance. The spatial mapping of these affordances returned a number of fuzzy zones indicating varying degrees of acceptability and tolerance for activities relating to the class of each element, Figure 15, left. These can be linked to quantitative analyses of e.g. daylighting or ventilation. Qualitative aspects are expressed in relational terms (e.g. view as visual access to windows).

Space affordances derived jointly from programmatic requirements and general (or alternative) uses of the spaces. This allowed for a combination of the accommodation of the activities in the brief and general concerns with space use and quality. The resulting functional and spatial patterns were also mapped as local coordinating devices that represented fuzzy zones but without a precise focus on building elements. Instead, they were bounded by the limits of spatial entities (generally individual spaces but possibly also wings or whole floors) and linked to the affordances of building elements on the basis of programmatic or general requirements, Figure 15, right. This meant that e.g. a workplace was linked to a window for daylight and view, to walls for acoustic isolation and to a door for pedestrian access for communication with other activities in the building or fire safety.

The links between spaces, activities and were generally sufficient for making explicit the potential of a design solution with respect to particular aspects from early on. In the example of Figure 15 the combined affordances of critical building elements defined areas amenable to different activities (left). The accommodation of required activities within these areas revealed a preference for particular workplace orientations, which in turn determined the spatial arrangement of workplaces and provided feedback to the brief with respect to the clustering of workplaces and the expectations concerning the size and facilities of spaces that accommodated the workplaces.

Adding affordances to design representations as abstract coordinating devices proved an interesting alternative to realistic simulations of use,

especially those involving virtual users. The spatial zones and functional relationships returned by affordance mapping were more economical in terms of time and computation and provided a transparent overview of possibilities and limitations. Affordances could also be complementary to user simulation, as they can guide interaction of autonomous virtual users with the building to the zones of interest or critical areas and thus reduce the number of iterations necessary for an adequate analysis.

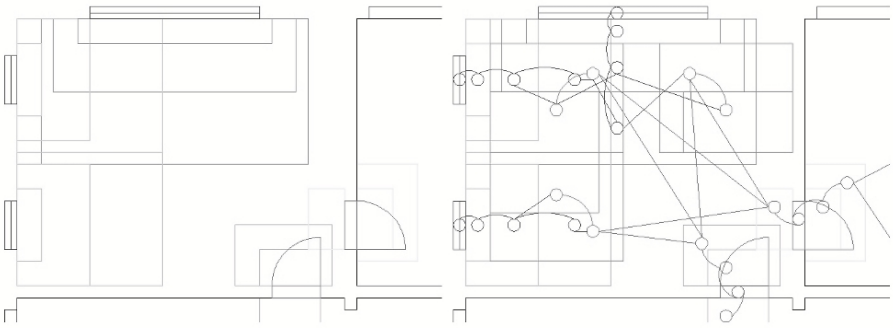


Figure 15. Affordance zones of openings (left) and correlation with affordance zones of activities (right).

The main prerequisite to the integration of affordances in design representations and analyses of especially programmatic requirements is the development of an extensive repertory of affordance definitions which express the different priorities and capabilities of a wide variety of users under different conditions. As the spatial representation of affordances can be cumulative, these variations can be accommodated in a few definitions which can be constantly augmented and refined with the viewpoints of different actors (e.g. sitting affordances at a desk could be enriched with the viewpoint of wheelchair users). In other words, affordances should not be exclusive or selective but inclusive and comprehensive. This may initially reduce apparent consistency but as understanding of the structure of underlying constraints and dimensions improves we are increasingly capable of identifying fundamental common elements.

6. Discussion

The use of affordances in architecture promises a compact, direct and transparent treatment of functionality and usability, which moreover agrees with the architects' intuitive handling of such issues. In deciding on the form and size of a space or the way users could move from one space to another, architects arguably make use of affordances rather than extensive and

detailed analyses to arrive at a satisfactory solution. The main advantage of affordances lies in the integration of information concerning functionality and usability into comprehensive structures which can be applied throughout the life cycle of a building. This should facilitate continuity of functional criteria and a better understanding of building performance.

It should be stressed that the main target of affordances in architectural design is the enrichment of the architects' perception. Use of affordances to guide use through design so as to reduce error margins is probably too deterministic for most uses of built environment beyond direct interaction with a building element. Moreover, buildings are tolerant to user errors. Taking the wrong route in a complex building can add to the inefficiency of pedestrian circulation, disorientation etc. but is not critical, unless under extreme conditions (e.g. fire escape). Other than encouraging errors and causing mild irritation, most misaffordances pose few dangers for the user outside such critical conditions (Evans and Mitchell McCoy 1998). Affordances are more important for design guidance through understanding, i.e. the combination of (a) the precision and independence required by objective analysis and (b) the subjective, meaningful qualities of human experience (Heft 1997). Unfortunately conflicts between constructivist and positivist positions tend to confuse the role of affordances in this combination (Oliver 2005).

A recent comparison between functions and the affordances in designing (Brown and Blessing 2005) suggests that affordances are primarily applicable once a conceptual design has been developed. Prior to that functional reasoning provides a higher focus on the goals and intentions of the design. This conclusion is consistent with emphasis on the intended function of a design that probably characterizes the majority of design and engineering disciplines. Architecture is arguably less successful with the sharp definition of intended functions, presumably because of the complexity of human activities in the built environment. The brief of a building is inevitably a very partial and elliptical document that stresses particular aspects while assuming substantial levels of complementary common sense and professional knowledge. Moreover, users of the built environment are particularly skilled at bypassing intended function without altering the form of a designed object, as in Figure 11. Consequently, in contrast to Brown and Blessing (2005) I propose that affordances are more than an addition to functional reasoning in building design: affordances are capable of integrating the unintended with the intended in general representations of functionality, usability and performance that also allow for direct and objective analysis and evaluation.

A prerequisite to achieving design guidance through affordances is the correlation of perceptions of the different parties involved in a building. POET stresses that designers and users have different conceptual models,

which communicate only indirectly through the system image (the physical image built on the basis of the designer's specifications, complemented with use indications such as documentation). To these two parties we should also add clients and authorities, with their own particular conceptual models and reference frames. At a basic level the affordances perceived by all these parties are common and derive from everyday use of the built environment. Distortion comes from differences in priorities and related semantic and cultural constraints. These differences are less pronounced in the perceived affordances of building elements, even though knowledge of architecture and building helps explain several constraints and adds more clues. The perception of space affordances is the weak point of most users (who may rely too much on trial and error), clients and authorities (who may be too selective) but also of architects (who may rely on stereotypical user profiles and be unable or unwilling to communicate and serve use). All parties may also suffer from false causality, i.e. coincidences. Such problems often lead to stereotypes and misconceptions due to limitations of the common sense users and clients rely upon or to professional/scientific assumptions that designers use to simplify problems, even when they conflict with everyday experience.

One of the principal contributions of affordances to architectural design is the potential ability to understand and utilize different aspects of users, including different degrees of mobility, perceptual or cognitive capabilities. By studying the affordances that relate such aspects and the resulting types to building elements and spaces, architects can go beyond vague, stereotypical user profiles, gross generalizations and arbitrary selections. The resulting insights should lead not to deterministic design solutions but to better understanding of space as a flexible and adaptable arrangement of multiple, overlapping opportunities (as opposed to the adjustability of a mono-functional object like a bicycle). Designers should be able to develop and communicate such opportunities through transparent devices such as feedback. Unfortunately buildings are not explicitly designed to provide feedback in the same way that a telephone button gives tactile or auditory feedback. Elements such as doors have locks that click but spaces have no feedback means other than potentially harmful conflicts (e.g. bumping one's head on a low beam). Spatial inconvenience tends to be mild (e.g. limited leg space) and may only become obvious over time.

From a technical point of view, the most striking aspect of affordances is mapping and in particular the selectivity of mapping. Affordances involve a direct correlation of user functions with objects that should be of interest to architecture. The immediacy of matching for example the whole hand in a particular orientation to a door handle, a finger to a button and the thumb and index finger to a key is not simply a matter of experience but also involves complex cognitive processing of form and scale. This could address

some of the fundamental weaknesses in architectural analysis, like resolution limitations due to normative thinking, and relates to the use of variable resolution and abstraction in design and affordance representations, such as multilevel, modular hierarchical representations (Marr 1982; Rosenfeld 1984; 1990). In computer vision these support simultaneous attention for e.g. different parts of a person's anatomy at various levels of abstraction. In architectural design they could support similarly simultaneous treatment of abstract entities, relationships and critical details, such as the inclusion of interfaces like door handles in early, abstract representations of doors.

References

Biederman, I: 1987, Recognition-by-components: A theory of human image understanding, *Psychological Review* **94**(2): 115-147.

Brown, DC and Blessing, L: 2005, The relationship between function and affordance, *Proceedings of IDETC/CIE: ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Long Beach, California.

Evans, GW and Mitchell McCoy, J: 1998, When buildings don't work: The role of architecture in human health, *Journal of Experimental Psychology* **18**: 85-94.

Gibson, EJ and Walk, RD: 1960, The "Visual Cliff", *Scientific American* **202**: 64-71.

Gibson, JJ: 1977, The theory of affordances, *in* RE Shaw and J Bransford (eds), *Perceiving, Acting and Knowing*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, pp. 67-82.

Gibson, JJ: 1979, *The Ecological Approach to Visual Perception*, Houghton Mifflin, Boston, Massachusetts.

Heft, H: 1997, The relevance of Gibson's ecological approach to perception for environment-behavior studies, *in* GT Moore and RW Marans (eds), *Advances in Environment, Behavior, and Design, Vol. 4*, Plenum, New York, pp. 72-108.

Hoffman, DD and Richards, W: 1985, Parts of recognition, *Cognition* **18**: 65-96.

Ingold, T: 1992, Culture and the perception of the environment, *in* E Croll and D Parkin (eds), *Bush Base: Forest Farm - Culture, Environment and Development*, Routledge, London, pp. 39-56.

Kim, HS, Park, KH and Kim, M: 1987, Shape decomposition by colinearity, *Pattern Recognition Letters* **6**: 335-340.

Koutamanis, A: 1997, Multilevel representation of architectural designs, *in* R Coyne, M Ramscar, J Lee and K Zreik (eds), *Design and the Net*, Europia Productions, Paris, pp. 74-84.

Koutamanis, A, van Leusen, M and Mitossi, V: 2001, Route analysis in complex buildings, *in* B de Vries, J van Leeuwen and H Achten (eds), *Computer Aided Architectural Design Futures 2001*, Kluwer, Dordrecht, pp. 711-724.

Marr, D: 1982, *Computer Vision*, WH Freeman, San Francisco.

McGrenere, J and Ho, W: 2000, Affordances: Clarifying and evolving a concept, *in Proceedings of Graphics Interface 2000*, Montreal, pp. 179-186.

Norman, D: 1988, *The Psychology of Everyday Things*, Basic Books, New York.

Norman, D: 2002, *The Design of Everyday Things*, Basic Books, New York.

Norman, DA: 1999, Affordance, conventions, and design, *ACM Interactions* **6**(3): 38-43.

Oliver, M: 2005, The problem with affordance, *E-Learning* **12**(4): 402-413.

Rosenfeld, A (ed): 1984, *Multiresolution Image Processing and Analysis*, Springer, Berlin.

Rosenfeld, A: 1990, Pyramid algorithms for efficient vision, *in* C Blakemore (ed), *Vision: Coding and Efficiency*, Cambridge University Press, Cambridge, pp. 423-430.

Tweed, C: 2001, Highlighting the affordances of designs, *in* B de Vries, J van Leeuwen and H Achten (eds), *Computer Aided Architectural Design Futures 2001*, Kluwer, Dordrecht, pp. 681-696.

THE ROLE OF PRECONCEPTIONS IN DESIGN

Some implications for the development of computational design tools

PATRICK HT JANSSEN
University of Melbourne, Australia

Abstract. The preconceptions that designers bring to the table when they are considering a particular design task are an unavoidable and necessary part of the design process. This paper first reviews the literature relating to the role of preconceptions in design, and then goes on to discuss computational tools that support the development and expression of such design preconceptions. As an example of such a tool, an outline is given of a generative evolutionary design system that allows designers to evolve families of designs that embody preconceived values and ideas.

1. Introduction

Designing will never be an entirely rational process. Assuming that the design task being undertaken is not trivial, then there will always be a complex process mediating between the initial design brief and the final design.

Much of the research into design methods during the 1960s hoped to create a design process that would somehow exclude all forms of subjectivity, thereby disallowing any personal creativity and intuition. The idea was that if the analysis of the 'problem' were sufficiently thorough, the 'solution' would emerge directly from the analysis. Furthermore, an objective procedure was sought that prescribed exactly how such a thorough analysis might be performed (Cross 1984).

Today, notions of creating 'optimal' designs are no longer realistic. Design problems are generally accepted as being highly 'ill-defined' or 'wicked' (Churchman 1967; Simon 1969; Rittel 1973). An ill-defined or wicked problem is one in which the requirements do not provide sufficient information to enable a solution to be found. Such problems require additional information to be discovered, created and invented. Design problems, like most everyday problems, tend to be ill-defined in an extreme

J.S. Gero (ed.), Design Computing and Cognition '06, 365–383.
© 2006 *Springer. Printed in the Netherlands.*

way in that the additional information is far greater that the information contained in the stated requirements.

Various researchers have argued that the missing inputs into the design process are the designer's *preconceptions*. Broadbent (1988) writes: "The presumption behind much theorising of the 60's was that *somehow* the Process would generate the form. But it rarely happened like that for the obvious reason that most architects approach most of their designing with certain preconceptions concerning, not just the *partie* of the building type in question but also, specifically, the *style*."

This paper first discusses the literature relating to the role of preconceptions in design, and then goes on to discuss computational tools that support the development and expression of such design preconceptions.

2. The Design Process Revisited

The typical 1960's design process consisted of five stages: briefing, analysis, synthesis, evaluation, and implementation. Analysis, synthesis, and evaluation were the three core stages that related directly to the process of designing and were generally assumed to be applied multiple times at different scales.

However, even in the 60's and 70's, researchers already acknowledged that the synthesis stage could not rely purely on the analysis stage. For example, Jones (1970) writes that the synthesis stages "is the stage when judgements and values, as well as technicalities, are combined in decisions that reflect the political, economic and operational realities of the design situation. Out of all this comes the general character, or pattern, of what is being designed, a pattern that is perceived as appropriate but cannot be proved to be right".

2.1. DESIGN PRECONCEPTIONS

The reference to 'judgements and values' suggests that some additional kind of input is required from the designer. But it is not clear what kind of input this might be. The methods discussed by Jones indicate that this input focuses primarily on redefining the problem. However, in areas such as architecture, the input from the designer will be much more dramatic and substantial.

2.1.1. The role of preconceptions

Broadbent (1988) modifies the 1960's design process by inserting 'preconception' as an extra input into 'synthesis', at right angles to the flow, Figure 1. In addition, he also adds a link from analysis to evaluation. Broadbent writes: "Different architects… will, given the same brief and even the same analysis come up with quite a different syntheses: Modern, Post-

Modern or whatever... In other words *whatever* kind of analysis has been brought to bear that architect, conditioned by the 'paradigm' in which he works, will bring in *sideways* to the Process the kind of design he wanted to do anyway!"

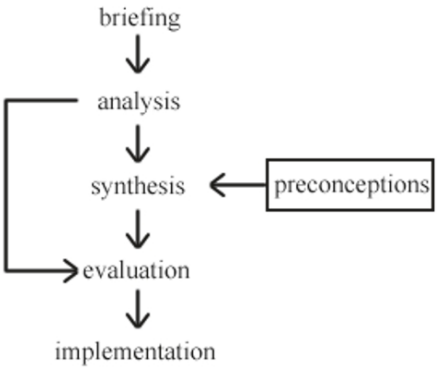


Figure 1. Broadbent's adaptation of the typical 1960's design process.

The importance of the preconceptions of the designer is confirmed when the work of most designer is analysed. In nearly all cases, the same set of beliefs and values may be identified in all the designs, with some designers repeatedly exploring the same specific design ideas. For example, in an interview, Frank Gehry describes his design process as follows:

> "I was not conscious that it (the Bilbao Guggenheim) had something to do with what I did before until later because you know, I'm just looking at what I see. I tend to live in the present, and what I see is what I do. And what I do is react. Then I realise that I did it before. I think it is like that because you can't escape your own language. How many things can you really invent in your lifetime. You bring to the table certain things. What's exciting, you tweak them based on the context and the people" (Bruggen 1997, p. 33).

2.1.2. Types of preconceptions

Within the literature, two types of preconceptions can be identified: a general *design stance* and a specific set of *design ideas*.

The design stance encompasses the broad beliefs and values that a designer holds. Designer may have the same design stance throughout their lifetime. The design ideas are related to the design stance, but are much more specific to particular types of project contexts and project constraints.

2.2. DESIGN STANCE

Three versions of the design stance are introduced: Broadbent's *paradigmatic stance*, Lawson's *guiding principles*, and Rowe's *theoretical position*.

2.2.1. Broadbent's 'paradigmatic stance'

Broadbent (1988) describes general types of preconceptions as the designer's *paradigmatic stance*. Such a stance may have been developed over a number of decades and may encompass broad philosophical beliefs, cultural values and perhaps some whimsical tendencies. A client's decision to employ a particular design team is likely to be based – at least in part – on the paradigmatic design stance of the design team in question.

2.2.2. Lawson's 'guiding principles'

Lawson describes this design stance as a set of *guiding principles*.

> "The designer does not approach each design problem afresh with a *tabula rasa*, or a blank mind, as is implied by a considerable amount of the literature on design methods. Rather, designers have their own motivations, reasons for wanting to design, sets of beliefs, values, and attitudes. In particular, designers usually develop quite strong sets of views about the way design in their field should be practiced. This intellectual baggage is brought by the designer into each project, sometimes consciously and at other times rather less so... Whether they represent a collection of disjointed ideas, a coherent philosophy or even a complete theory of design, these ideas can be seen as a set of 'guiding principles'." (Lawson 1997, p. 162)

Lawson describes how these guiding principles may differ in content from one designer to the next, and also how they are used in different ways. In terms of content, Lawson lists six types of constraints that play an important part in defining the guiding principles of a designer: client constraints, user constraints, practical constraints, radical constraints, formal constraints and symbolic constraints. The attitude of the designer towards these constraints, may to a large extent define their guiding principles.

2.2.3. Rowe's 'theoretical position'

Rowe (1987) has developed a similar concept, that he refers to as a designers *theoretical position*.

The theoretical position consists of a set of general arguments and principles that may be either explicit or implicit. In some cases, the theoretical position is well defined, in other cases it is more implicit and vague. Most theoretical positions in some way attempt to address the question: "What is proper architecture?".

Rowe creates an analytical framework to describe and compare different types of theoretical position. The framework describes a theoretical position as a three stage argument that progresses from general statements to specific types of architecture. The three stages are labelled as *orientation*, *architectural devices*, and *production*. These stages are described as follows:

- Orientation covers the critical stance and larger purpose of the position.
- Architectural devices refers to architectonic elements and leitmotifs (for example, Le Corbusier's five points) that describe the position's production.
- Production describe a family of buildings identified by some label ('brutalist', for example).

According to Rowe, a theoretical position can be characterised as an argument that first sets out a broad orientation, this orientation will support certain architectural devices, and these devices will in turn lead to certain types of architectural production.

2.3. DESIGN IDEAS

As well as the overall design stance, designers will generally tackle a particular design task with certain initial design ideas. These ideas will be compatible with their design stance, but will be much more specific to the design task being considered. They reflect the design stance combined with a 'gut feeling' inspiration on how to approach the design task.

As with the design stance, researchers have proposed various versions of such design ideas. Three versions will be discussed: Darke and Lawson's *primary generators*, Rowe's *enabling prejudices*, and Frazer's *working methods*.

2.3.1. Darke and Lawson's 'primary generators'

Darke (1979) conducted a series of interviews with British architects about their intentions when designing local authority housing. Through these interviews, Darke highlights how many architects (but not necessarily all architects) latch onto a relatively simple set of related concepts and ideas early on in the design process. Darke refers to this concept as a *primary generator*.

Concerning such primary generators, Darke emphasises three key points:

- They are developed early on in the design process, prior to a detailed analysis of the design problem.
- They are not created by a process of reasoning, but are instead an "article of faith on the part of the architect".
- They provide a framework that defines and directs the overall design approach. In particular, the primary generators structure the problem definition, rather than vice versa.

Darke concludes that early on in the design process, designers "fix on a particular objective, or small group of objectives, usually strongly valued and self imposed, for reasons that rest on their subjective judgement rather than being reached by a process of logic".

Lawson (1997) describes a series of protocol studies of design exercises supporting the conclusions reached by Darke (see Eastman (1970) and Agabani (1980)). Lawson (1997) also emphasises the importance of these

design ideas in the overall design process. In addition, Lawson stresses that the number of primary generators may be small. Lawson (1997) writes: "Good design often seems to have only a few major dominating ideas which structure the scheme and around which the minor considerations are organised. Sometimes they can be reduced to only one main idea known to designers by many names but most often called the 'concept' or the 'parti'".

2.3.2. Rowe's 'enabling prejudices'

Further evidence supporting the idea of the primary generator has also been collected by Rowe, using protocol studies and analysis of written sources. Three case studies of designers in action were analysed, and an attempt was made to reconstruct the sequence of steps, moves and other procedures used. In addition, further examples of the design process, taken from various written sources, were also analysed.

One of the key characteristics to emerge was the dominance of initial design ideas on the rest of the design process. Rowe (1987) writes: "Initial design ideas appropriated from outside the immediate context of a specific problem are often highly influential in the making of design proposals. Quite often references are made to objects already in the domain of architecture. On other occasions, however, an analogy is made with objects and organizational concepts that are further afield and outside architecture". Rowe refers to these initial ideas and references as *enabling prejudices*.

Based on the case studies and written sources, Rowe emphasises two key points about such enabling prejudices:

- They are more important than the problem conditions and tend to be the driving force behind the whole design process.
- In many cases, they are not discarded at the end of a project, but instead become long-lasting themes explored through multiple projects.

2.3.3. Frazer's 'working methods'

Frazer describes a general design methodology common to many design fields that develop design ideas through multiple projects (Frazer 1974; 2002; Frazer and Connor 1979).

He conceptualises the design ideas as being embedded in the *working methods* of designers, and that it is these methods that "characterise their 'style'". Furthermore, Frazer highlights how aspects of these methods are explicitly defined in many offices as standard details, templates, procedures, and so forth. He describes this methodology as both personal because it is particular to one designer, and generic because this designer will use it in multiple projects. Frazer writes:

> "It is common to find sets of standard details in architects' offices that serve to economise in time, ensure details are well tested, but also to ensure a consistency of detailing and to reinforce the house style. In many offices this

> extends to design procedures, approaches to organization and so forth. The same is true of industrial designers where again stylistic characteristics such as details, colour, preferred materials give economy, consistency, quality control and identifiable house style... The identifying characteristics often go through changes during the development of the designers, sometimes with abrupt changes as with Le Corbusier, but usually a continuous progression can be seen. The stylistic characteristics can continue with an office, studio or company, long after the death of the original designer."

Frazer makes two important points about design ideas:

- They are developed on a long-term basis through multiple projects.
- They are embedded in the practical working methods used by designers, encompassing procedures, tools and data.

2.4. CONTINUITY BETWEEN DESIGNS

The design process described by Broadbent, Figure 1, can now be further modified to incorporate a number of concepts from the frameworks described above.

In particular, the process should reflect the fact that a designer's body of work does not consist of a disparate set of unrelated designs. Instead, the individual designs tend to be related to each other, and can be seen to reflect a design stance, as well as specific design ideas. The individual designs may often be seen as part of a personal process of exploration and development. The designs may form a stylistic family or a chronological sequence, or parts of one design may be found in another design. As a result of the interrelationships between the designs, each design becomes recognisable as being part of the designer's body of work.

In order to capture the continuity between designs, a conceptual entity is proposed that represents the character of a varied family of designs by one design team. This entity is referred to as a *design schema* (Janssen et al. 2000), and will consist of an interrelated set of design ideas that are compatible with a particular design stance.

2.4.1. The design schema

The term 'schema' is widely used in cognitive psychology and the cognitive sciences generally to designate a mental structure that encapsulates generic knowledge about types of events, objects or actions. The term is generally attributed to Frederic Bartlett (1932), as part of his work on constructive memory. Following the suggestion of his neurologist friend Henry Head, Bartlett argued that memories were not stored as static structures waiting to be revived, but instead formed parts of large complexes, called *schemata*, that could be modified whenever they were retrieved.

For example, a schema may exist that encapsulates general knowledge required to walk down a staircase. This staircase-descent schema is generic

in the sense that it can be applied to a wide variety of staircases. When the task of descending the staircase is about to be performed, the relevant schema will be retrieved from long-term memory and, if necessary, adapted to the features of the specific staircase. Such schemas are acquired and developed over a lifetime of learning.

A design schema is similar, although much more complex. Each time a designer is faced with a design project, they will retrieve from long-term memory their schema that most closely fits the task at hand. The nature of such a schema will depend on the designer in question, but it will tend to consist of a mixture of both declarative and procedural patterns. Declarative patterns describe aspects of the design product, while procedural patterns describe aspects of the design process. Such patterns will reflect the design stance and will embody a set of design ideas, in most cases in a highly implicit way.

If necessary, this schema may be adapted to better fit the specific design task. A design team will tend to develop numerous design schemas for different types of projects. Each schema is then applied to numerous projects, and may be adapted and modified each time it is used. Since these schemas belong to a single design team, they will generally embody the same design stance, but the design ideas are likely to vary depending on the type of project for which they have been developed. The use and retrieval of schemas by designers is also discussed by Lawson (2004).

2.4.2. The design environment

The concept of the design schema results in a more complex relationship with the context and constraints of a particular project. Collectively, the context and constraints are referred to as the *design environment*, and will typically include the site, the brief or programme, the budget, performance targets, and so forth.

A design schema will be generic in the sense that it will not be specific to one design environment. Instead, the design team develops it with a certain *type* of environment in mind, referred to as a *niche environment*, encompassing a range of possible contexts and a range of possible constraints. The schema can be used in any project whose design environment matches the niche environment for which the schema was developed.

2.4.3. Character versus configuration

The design ideas embedded in a schema's patterns result in certain characteristics common to all members of the family. Such characteristics will relate to a complete spectrum of design considerations, including tectonic, functional and formal issues.

With regard to these shared characteristics, a distinction may be made between the design *character* and the design *configuration*.

- Character refers to the apparent distinctive and recognisable nature of the design. It is achieved by triggering an emotional response and is therefore seen as being highly subjective.
- Configuration refers to the selection, relationship and arrangement of parts and components of the design. It defines the specific topological and geometrical organisation of the overall design, and is therefore seen as being highly objective.

Within any family of designs, there will clearly be a considerable amount of interdependence between character and configuration. Character may limit possible configurations, and configuration may influence the resulting character. However, these two qualities are nevertheless seen to be reasonably distinct, in that when one is kept constant, the other may still vary significantly. Designs may share the same character but nevertheless differ in their configuration. Or inversely, designs with identical configurations may have completely different characters.

In order for the schema to remain applicable to a wide range of projects, the configuration of the design must remain flexible so as to allow the schema to be adapted to differing constraints and contexts. Design schemas therefore tend to predefine the design character, but refrain from defining the design configuration.

For a particular designer, the character of the schema should not be thought of as a fanciful choice, in the sense that they may choose one style or another. Rather, the character of their designs is inherent in the way that they operate, and often constitutes a set of strongly held beliefs. The term ‘style’ may therefore not be appropriate (Lawson 1997, p. 163-166). Design style tends to focus on formal issues, whereas design character involves a much wider range of issues, including functional and tectonic issues.

2.4.4. Exploration of alternative configurations

When a design schema is actually applied within a project, the patterns from the schema may first be adapted to fit the specific environment for the project. Once a satisficing match between the schema and environment is achieved, a variety of alternative configurations may be synthesised. The resulting designs will vary in configuration, but will embody the same design stance and the same set of design ideas.

The design alternatives will then need to be evaluated. Some configurations will be seen to be more desirable than others. Such evaluations are likely to be based on both qualitative judgements such as design elegance, on quantitative judgements such as design performance. The synthesis and evaluation of alternative design is often referred to as a process of exploration (Smithers et al. 1994).

The design schema will have embedded within it the approach to be used to evaluate designs. For example, as part of the design stance and the design ideas, one schema may prioritise energy efficiency, while another might focus on low budget construction. Designs are therefore not evaluated in absolute terms, but instead are only evaluated relative to other designs from the same schema.

The design schema therefore defines its own value system. This fits with the idea that a designers to a large extent define the problems that they aim to solve, also referred to as problem structuring (Archer 1979). By defining its own evaluation approach, a design schema defines which basic problems will be tackled. The brief or programme for a specific project may then provide further detailed information for these problems.

2.4.5. Schema based design process

A model of a design process is proposed based on the concept of design schemas. This design process is shown in Figure 2. This process, although not necessarily universal, is seen as one that many designers loosely follow.

The design process consists of three stages: the *character cultivation* stage, the *configuration exploration* stage, and *detail specification* stage.

- In the character cultivation stage, a design schema is developed that consists of an interrelated set of design ideas applicable to certain types of projects. The main guiding forces at this stage are the niche environment and the design stance of the design team.
- In the configuration exploration stage, alternative design configurations are explored by applying the design schema to the constraints and context for a specific project. This stage involves analysing the brief and the site, and exercising the design ideas in an appropriate manner.
- In the detail specification stage, the design team further develops the design model selected in the previous stage to the level of detail required for construction.

The first stage constitutes the schema development phase, which tends to unfold over an extended period of time, and may even develop in a highly subconscious way. Designers will usually develop such ideas by working on a series of similar projects or competitions. The second and third stages constitute the design development phase, which is much more deliberate and tends to be much shorter as it is limited to the lifespan of a single project.

Although the direction of the flow should be understood to be predominantly top to bottom, most designers do not work in a linear manner. The schema based design process is loaded with the potential for conflicts, ambiguities, and misunderstandings that are typically resolved through an iterative process that moves back and forth between different team members, software systems, and design stages.

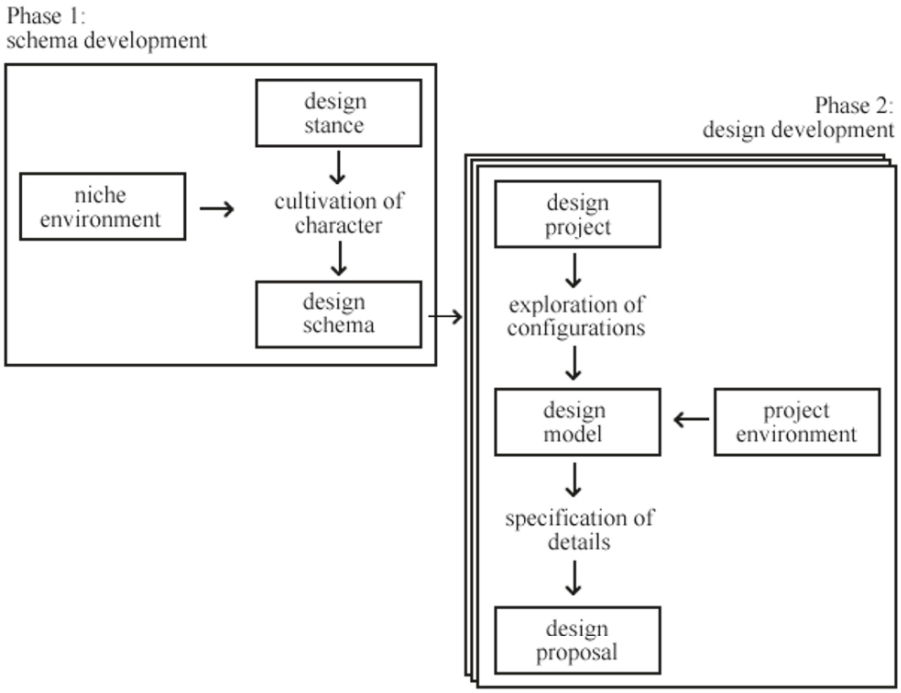


Figure 2. A design process that many designers follow.

2.4.6. A note on design preconceptions

Generally in the design field there is a high level of prejudice against any kind of design preconceptions. They are often considered something to be avoided at all costs – design quality is considered to be the result of an innovative mind that rejects conventional preconceptions in favour of the new and more radical ideas. However, the frameworks described above, as well as the schema based design process, suggest that these types of preconceptions are a necessary part of the design process, and furthermore that they can be a positive and enabling influence.

Preconceived design ideas should therefore be treated with caution: some may be *limiting* in that they seem to restrict the freedom of the designer, while others may be *enabling* in that they give the designer greater freedom.

Designers will generally try to discard preconceptions that they see as limiting, and develop those that are enabling. However, identifying which are limiting and which are enabling is not straightforward. Preconceptions not only affect the way a design is created, but also affect the perceived quality of a design. Preconceptions will affect the choice of objectives, the importance assigned to each objective, and the way that each objective is evaluated.

3. Some Implications for Computational Design Tools

In general, the schema based design process falls within what Mitchell (1994) has described as the *designing as social activity* paradigm. Human and computer agents each have their own (not necessarily consistent, comparable, or compatible) knowledge bases and problem-solving capabilities, and interact with one another over the network. They import knowledge into the common pool, they construct some common intellectual ground, and they attempt to form consensus and resolve problems.

One possible computer agent is a system to assist in the configuration exploration stage. If the design schema could be encoded in a format that could be used by a computer, then an automated exploration system might be able to use this encoded schema as a basis for exploring alternative design configurations. The design models that would then be generated would all reflect the same design stance and share the same set of design ideas.

Furthermore, the ability of human designers to explore and evaluate large numbers of alternative configurations is limited, due to factors such as boredom and fatigue. While a human designer may only consider a handful of options, a computer may be able to fully evaluate thousands of alternatives. The automation of this stage is therefore seen as being potentially highly beneficial, since it is relatively repetitive as numerous design alternatives are considered and evaluated.

3.1. A GENERATIVE EVOLUTIONARY DESIGN FRAMEWORK

A computational design system is proposed that aims to automate the design exploration stage of the schema based design process. This system uses generative and evolutionary algorithms to develop alternative design configurations.

3.1.1. The evolutionary approach

A wide variety of evolutionary algorithms exist, with the four main types being genetic algorithms (Holland 1975), evolution strategies (Rechenberg 1973), evolutionary programming (Fogel 1995), and genetic programming (Koza 1992). Such algorithms are loosely based on the neo-Darwinian model of evolution through natural selection. A population of individuals is maintained and an iterative process applies a number of evolution steps that create, transform, and delete individuals in the population.

With evolutionary design systems, each individual has a genotype representation and a phenotype representation. The genotype representation encodes information that can be used to create a model of the design, while the phenotype representation is the actual design model. The individuals in the population are rated for their effectiveness, and on the basis of these evaluations, new individuals are created using ‘genetic operators’ such as

crossover and mutation. The process is continued through a number of generations so as to ensure that the population as a whole evolves and adapts.

Generative evolutionary design systems require a generative process to generate alternative design models using the information stored in the genotype. This process consists of a rule-based growth procedure that is capable of generating design alternatives that vary significantly from one another. This approach may be used early on in the design process and focuses on the discovery of inspiring or challenging design alternatives for ill-defined design tasks. The evolutionary system will tend to evolve a divergent set of alternative designs, with convergence on a single design often being undesirable or even impossible. Such systems are sometimes described as 'divergent systems', 'exploration systems' or 'synthesis systems'. Examples of generative evolutionary design systems include (Frazer and Connor 1979; Graham et al. 1993; Frazer 1995; Bentley 1996; Rosenman 1996; Shea 1997; Coates et al. 1999; Funes and Pollack 1999; Sun 2001).

The proposed generative evolutionary design framework allows designers to incorporate and express their own design schemas (Janssen 2004; Janssen et al. 2005a; 2005b; 2006). This approach is based on the work of Frazer and Connor (1979), Frazer (1995), and Sun (2001). The proposed framework is broken down into two parts: a design method that sets out the main tasks to be performed by the design team, and a software system that allows the design team to evolve alternative design configurations.

3.1.2. Generative evolutionary design method

The proposed design method, Figure 3, is a modification of the existing schema based design process, shown in Figure 2. In the schema development phase, an additional stage has been added, referred to as the *rule formulation* stage, during which the schema is encoded as a set of evolutionary rules and representations. In the design development phase, the *configuration exploration* stage has been replaced by a *configuration evolution* stage.

- The character cultivation stage is similar to the existing design process and requires the design team to conceptually cultivate a set of design ideas for a particular niche environment. At this stage, the design team works at a purely conceptual level and as a result, no specialised programming or computational skills are required.
- In the rule formulation stage, the design schema is encoded in a form that can be used by the evolutionary system. This involves formulating a set of small programs called routines for the evolutionary system. The requirements from the design team are now different, and specialised programming and computational skills are essential.

- In the configuration evolution stage, a variety of alternative design configurations are evolved and adapted to a specific design environment. This stage therefore requires a software system to evolve alternative designs. The design team will need to encode the design environment for the specific project, and will then need to configure and run the evolutionary system.
- The detail specification stage is identical to the existing design process, with the selected design model being developed into a detailed design. As with the first stage, no specialised programming or computational skills are required.

The routines developed during the rule formulation stage will be used by the evolutionary system to execute four critical evolutionary steps: *reproduction*, *development*, *evaluation* and *survival*. Routines will also be required for *initialisation*, *visualisation* and *termination*. The design team will need to develop these routines so that the evolutionary system can be used to evolve design alternatives that reflect their design stance and that embody their design ideas. The rules and representations used in these routines will tend to represent the design stance and the design ideas in a highly implicit manner.

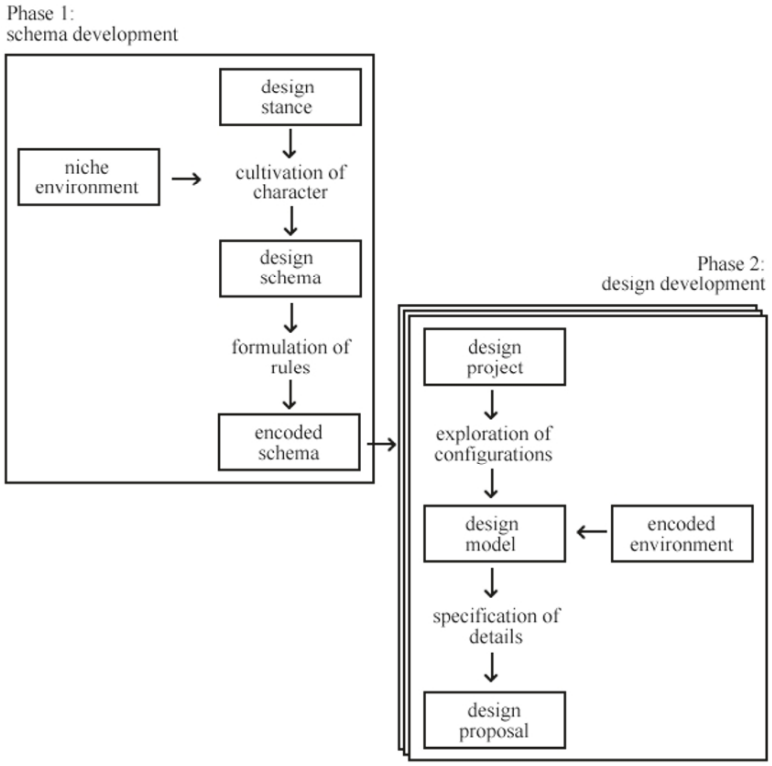


Figure 3. A generative evolutionary design method.

3.2. DEMONSTRATION

The evolutionary system is currently under development. In order to verify the feasibility of this approach, the rule formulation stage of the proposed design method has been tested and demonstrated. This involves the process of encoding a design schema as a set of routines. An example design schema for a family of multi-story buildings was defined, and a set of routines was then developed to encode this schema. Figure 4 shows a small selection of designs generated using these routines.

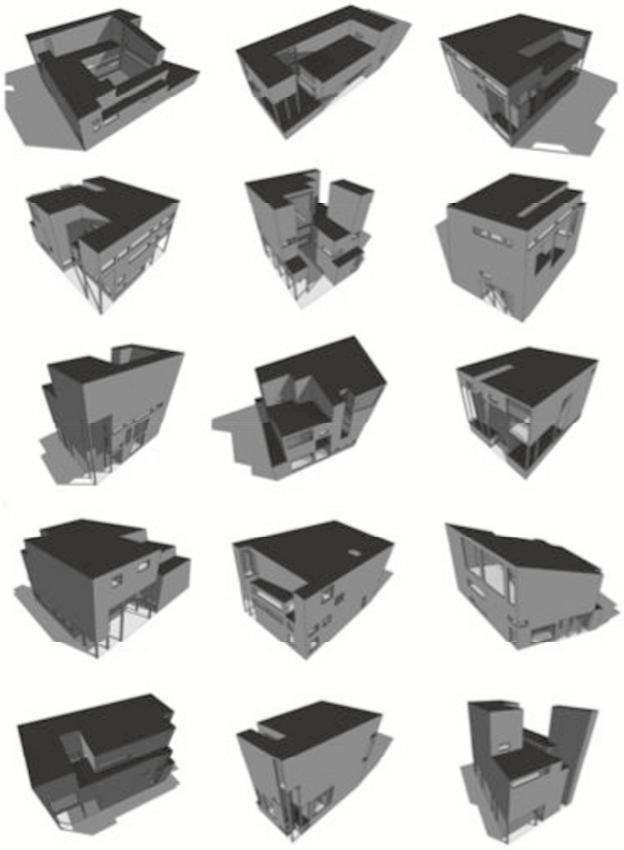


Figure 4. A set of generated (but not evolved) designs.

Three routines were implemented: initialisation, development and visualisation. The initialisation routine was used to generate a population of genotypes, the developmental routine was then used to generate a population of design models, and finally the visualisation routine was use to view these models.

The routines were implemented as stand-alone Java programs. The individuals were represented using XML, with three main nodes: 'genotype', 'phenotype' and 'evaluations'. The genotype node consisted of a set of real numbers that were the genes for the generative process. If the individual had been developed, then the phenotype contained the representation of the model, encoded using an XML representation. Finally, the evaluations node could contain one or more evaluation scores, each of which encoded the result of analysing one objective. Since the evaluation routines were not implemented in this demonstration, this part was not used.

3.2.1. Developmental routine

A hybrid generative process for the example schema has been developed, where a variety of generative techniques are sequentially applied to a grid generated using a simple spatial partitioning technique. The process consists of a sequence of eight steps that gradually change an orthogonal grid into a three-dimensional building model. The grid consists of as a set of intersecting faces that subdivide the space into an array of cells. Initially, the faces are orthogonal to one another and regularly spaced, creating an array of cubic cells.

The eight generative transformations are shown (diagrammatically in two-dimensions) in Figure 4: positioning of the grid in the site, translation of the grid-faces, inclination of outer grid-faces, insertion of the staircase, creation of spaces, selection of outside spaces, insertion of doors, and insertion of windows.

Most transformations require a set of parameters encoded within the genotype. The genotype consists of a fixed length list of real numbers in the range 0.0 to 1.0. Each number is referred to as a gene. The genes may be mapped to a value within a different continuous or discrete range as required. Some transformations may also require certain parameters or data encoded in data-files that describe the environmental context and constraints.

When executing the generative steps, the generative process must verify than certain hard constraints are not violated. For example, the constraints specify that all spaces must be higher than a certain minimum height, must have a horizontal floor level, and must be accessible from the staircase. In addition, a soft constraint specifies that the preferred spaces are rectangular shaped, followed by L-shaped spaces, followed by all other spaces (such as T-shaped spaces).

4. Conclusions

The demonstration focused on the rule formulation stage of the proposed generative evolutionary design method, Figure 3. This is the second stage of the schema development phase, and it was the stage that was inserted into the existing schema based design process, Figure 2. The encoding of the

design schema as a developmental routine was demonstrated. The routine generated alternative designs that differed significantly from one another in configuration, but that shared the same character, as shown in Figure 5. The rules and representations used in the generative process implicitly embodied this design character.

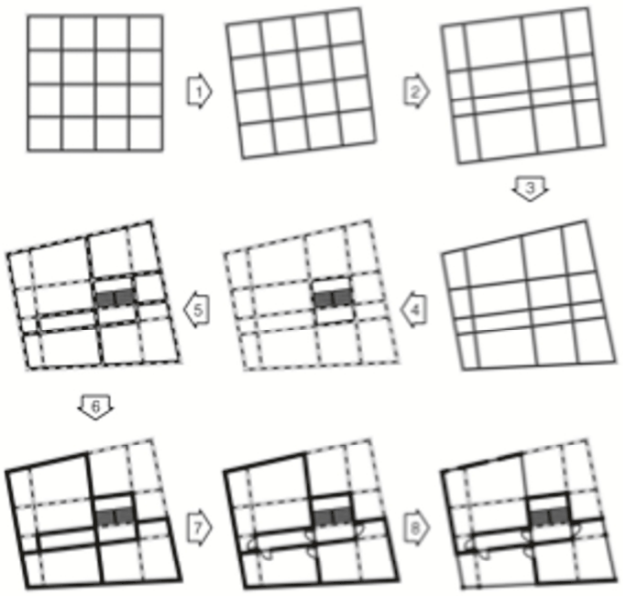


Figure 5. The generative process consisting of eight steps.

The next stage of the research will focus the configuration evolution stage. This stage requires an evolutionary system, which will allow the design team to evolve alternative designs in response to the environment.

A computational architecture has been developed for the system, to ensure that it will be both scalable and customisable (Janssen 2004). For scalability, the system uses a parallel client-server based architecture, whereby the most computationally expensive processes may be performed in parallel by multiple client computers. For customisability, the system uses a plug-in architecture, whereby specialised components (including the routines of the encoded schema) may be plugged into the underlying generic system. The evolutionary system is currently being implemented, which will allow further evaluations to be carried out.

References

Agabani, FA: 1980, *Cognitive Aspects in Architectural Design Problem Solving*, Doctoral Dissertation, University of Sheffield.

Archer, B: 1979. The three R's, *Design Studies* **1**(1): 18–20.

Bartlett, FC: 1932, *Remembering,* Cambridge University Press, Cambridge, UK.

Bentley, PJ: 1996, *Generic Evolutionary Design of Solid Objects using a Genetic Algorithm*, Doctoral Dissertation, Division of Computing and Control Systems, Department of Engineering, University of Huddersfield.

Broadbent, G. (1988). *Design in Architecture: Architecture and the Human Sciences*, David Fulton Publishers, London, UK.

Brown, DC: 1998, Defining configuring, *AI EDAM* **12**: 301-305.

Bruggen, CV: 1997, *Frank O Gehry: Guggenheim Museum Bilbao*. Harry N Abrams, New York, NY.

Churchman, CW: 1967, Wicked problems, *Management Science* **14**(4): B141-142.

Coates, P, Broughton, T, and Jackson, H: 1999, Exploring three-dimensional design worlds using Lindenmayer systems and genetic programming, *in* PJ Bentley (ed), *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco, CA, pp. 323-341.

Cross, N: 1984, *Developments in Design Methodology*, Wiley, Chichester.

Darke, J: 1979, The primary generator and the design process, *Design Studies* **1**(1): 36-44.

Eastman, CM: 1970, On the analysis of the intuitive design process, *in* GT Moore (ed), *Emerging Methods in Environmental Design and Planning*, MIT Press, Cambridge, MA, pp. 21-37.

Fogel, DB: 1995, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, IEEE Press.

Frazer, JH: 1974, Reptiles, *Architectural Design* **4**:231-239.

Frazer, JH: 1995, *An Evolutionary Architecture*, AA Publications, London, UK.

Frazer, JH: 2002, Creative design and the generative evolutionary paradigm, *in* PJ Bentley and DW Corne (eds), *Creative Evolutionary Systems*, Academic Press, London, UK, pp. 253-274.

Frazer, JH and Connor, J: 1979, A conceptual seeding technique for architectural design, *in International Conference on the Application of Computers in Architectural Design and Urban Planning*, Berlin, AMK, pp. 425-434.

Frazer, J: 1991, Can computers be just a tool? *Systemica: Mutual Uses of Cybernetics and Science* **9**: 27-36.

Funes, P and Pollack, J: 1999, Computer evolution of buildable objects, *in* PJ Bentley (ed), *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco, CA, pp. 387-403.

Graham, PC, Frazer, JH, and Hull, MC: 1993, The application of genetic algorithms to design problems with ill-defined or conflicting criteria, *in* R Glanville and G de Zeeuw (eds), *Proceedings of Conference on Values and, (In) Variants*, pp. 61-75.

Holland, JH: 1975, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Janssen, PHT: 2004, *A Design Method and a Computational Architecture for Generating and Evolving Building Designs*, Doctoral Dissertation, School of Design Hong Kong Polytechnic University.

Janssen, PHT, Frazer, JH and Tang, M-X: 2000, Evolutionary design systems: A conceptual framework for the creation of generative processes, *in Design Decision Support Systems in Architecture and Urban Planning*, Nijkerk, The Netherlands, pp. 190-200.

Janssen, PHT, Frazer, JH and Tang, M-X: 2005a, Generative evolutionary design: A system for generating and evolving three-dimensional building models, *Innovation in Architecture, Engineering and Construction*, Rotterdam, The Netherlands.

Janssen, PHT, Frazer, JH and Tang, M-X: 2005b, A computational system for generating and evolving building designs, *in Computer Aided Architectural Design Research in Asia,* pp. 463-474.

Janssen, PHT, Frazer, JH and Tang, M-X: 2006, A framework for generating and evolving building designs, *International Journal of Architectural Computing* **3**(4): 449-470.

Jones, JC: 1970), *Design Methods: Seeds of Human Futures*, Wiley, London, UK.

Koza, JR: 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA.

Lawson, B: 2004, Schemata, gambits and precedent: Some factors in design expertise, *Design Studies* **25**(5): 443-457.

Lawson, B: 1994 *Design in Mind*, Butterworth-Heinemann, Jordan Hill, Oxford, UK.

Lawson, B: 1997, *How Designers Think: The Design Process Demystified*. Architectural Press, Oxford, UK, 3rd ed.

Lawson, BR: 1972, *Problem Solving in Architectural Design*, Doctoral Dissertation, University of Aston, Birmingham.

Mitchell, WJ: 1994, Three paradigms for computer-aided design, *in* G Carrara and Y Kalay (eds), *Knowledge-Based Computer-Aided Architectural Design*, Elsevier, Amsterdam, pp. 379-388.

Rechenberg, I: 1973, *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog Verlag, Stuttgart, Germany.

Rittel, H: 1973, The state of the art in design methods, *Design Research and Methods (Design Methods and Theories)* **7**(2): 143-147.

Rosenman, MA:1996, An exploration into evolutionary models for non-routine design, *in AID'96 Workshop on Evolutionary Systems in Design*, pp. 33-38.

Rowe, PG: 1987, *Design Thinking*, MIT Press, Cambridge, MA.

Shea, K: 1997, *Essays of Discrete Structures: Purposeful Design of Grammatical Structures by Directed Stochastic Search*, Doctoral Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Simon, H: 1981, *The Sciences of the Artificial*, MIT Press, Cambridge, MA.

Smithers, T, Corne, D and Ross, P: 1994, On computing exploration and solving design problems, *in* JS Gero and E Tyugu, (eds), *Formal Design Methods for CAD*, North-Holland, Amsterdam, pp. 293-313.

Sun, J: 2001, *Application of Genetic Algorithms to Generative Product Design Support Systems*, Doctoral Dissertation, Hong Kong Polytechnic University.

HOW AM I DOING? THE LANGUAGE OF APPRAISAL IN DESIGN

ANDY DONG
University of Sydney, Australia

Abstract. Accounts of design practice suggest that how a designer 'feels' influences the way that a designer behaves. In this paper, we approach the notion of 'feeling' through the linguistic process of appraisal. Appraisal is the representation through language of favourable and unfavourable attitudes towards specific subjects. While the linguistic system of appraisal may not be completely isomorphic with the human system of emotions, characterising the language of emotions through a linguistic system allows us to begin an exploration of the affective aspects of the design process. The paper describes a taxonomy for a formal, grammatical analysis of appraisal of design processes, products and people. The taxonomy is used to analyze the way language is *structured* to adopt attitudinal stances in accounts of design experience. An analysis on design student blogs shows how the appraisal framework systematically accounts for the linguistic resources of appraisal in design and how appraisal inhabits design practice.

1. Introduction

1.1. AFFECT AND APPRAISAL

There is growing evidence that feeling, a common term to denote affect and emotion, influences cognition and is essential to rational, decision-analytic behaviour, learning, and, more generally, the human cognitive system. Forgas (2000), for example, claims that positive affect benefits creative activities such as design though designing was not explicitly studied. Affect is thought to arise from our appraisal of appetitive (positive) and aversive (negative) external stimuli. Emotion is the affective state resulting from the appetitive or aversive appraisal of external stimuli.

Feeling, however, is notably absent from the published literature on design cognition with a few exceptions (Love 1999, 2001). Various cognitive models of designers have been put forth, each of which attempt to provide explanatory frameworks for designers' mental representations and

J.S. Gero (ed.), Design Computing and Cognition '06, 385–404.
© 2006 *Springer. Printed in the Netherlands.*

reasoning processes; representational theories of the mind such as Newell's symbolic information processing model (Newell 1990) claim rules and representations of individuals' minds that enable them to solve problems. Connectionist models propose that design thinking processes could be modelled through implicit schemas which activate to form concepts. Regardless of the model of the internal mechanisms that account for design cognition, both approaches clearly exclude affect from their models. The cognitive paradigm is also characterised by a general lack of interest in affective influences on design cognition outside of prescribed experimental conditions.

Design researchers are just beginning to discover how emotions may directly influence design, such as having equivalent effect on ideation during conceptual design as formal reasoning (Solovyova 2003). This type of evidence reinforces cognitive science findings that affect influences human information processing. For example, Fiedler's BIAS model of affective cognition (Fiedler 2000) supports the assertion that positive affective states influence generative and creative thinking, whereas negative affective states promote detail and procedurally oriented thinking.

Studying emotions and design cognition empirically is problematic for various reasons. First, eliciting specific emotions from designers in a controlled laboratory situation by inducing stress or elation to compare their relative effects on design cognition raises serious ethical dilemmas. Second, emotions manifest in various and often ambiguous ways. Physiological expressions of emotions include facial expression (e.g., smiles) and body gestures (e.g., arms crossed) and are associated with a pattern of expression (e.g., a smile is generally an expression of happiness) (Ekman and Friesen 1975). Some of these patterns have been shown to be universal with respect to cultures (Ekman et al. 1987). Bio-physiological reactions to emotions include changes to blood pressure, skin conductivity, heart rate, and brain activity, all of which could be measured with appropriate machinery and correlated to various emotions.

Human language is vital to the expression of feelings and is comparatively easier to access than physiological or bio-physiological indicators of emotions. Language enables humans to express their conscious awareness of emotions; therefore, language allows others to assess the subjective feeling component of emotions. There has been detailed research into the language of emotions. Ortony (1987) established an "affective lexicon," a taxonomy that categorizes words about affect into affective, behavioural and cognitive "foci." The criterion for the placement of words in each of the foci is based upon the extent to which the word refers to how one is feeling (affect), acting (behaviour), or thinking (cognitive). Ortony (1987) distinguishes affect from emotion by classifying affect as relating to a valenced judgment whereas emotion is an affective state.

The available research in psychology (Pennebaker and King 1999) and linguistics points to systematic approaches to derive reliable linguistic resources that deal with both the representation and the representability of feelings through language. In linguistics, this process is called appraisal. Appraisal theory in linguistics deals with the way that speakers express evaluation, attitude and emotions through language. Martin (2000) reviews a wide array of linguistic resources deployed in appraisal and we refer the reader to his research in the basic linguistic resources for appraisal.

This paper discusses the language of appraisal for designers, and in particular, how designers appraise the process of their design practice, their product, the designed work, and how designers describe their own or other's affect and designerly capability. Thus, in accord with Ortony's distinction between affect and emotion, the language of appraisal deals with designers' valenced reaction to the design context or situation rather than the concomitant affective state. The broader aim of the research is to discover and systematize computationally a means to analyze a large body of design texts to discover the affective contents and processes that underlie, regulate and influence design cognition. Computationally assessing appraisal in text is known as sentiment analysis in computational linguistics (Turney and Littman 2003). A fundamental start to sentiment analysis is to establish the grammatical forms of sentiments and to what categories sentiment may belong – both of which will be accounted for in the framework for appraisal in design.

The framework for the language of appraisal presented in this paper is intended to analyze design text, that is, text that is written by designers to give an account of designing and the designed object. This paper modifies the appraisal framework proposed by Martin (2003) to take into account the ways that designers appraise. The first part of the paper describes the linguistic framework for the analysis of appraisal in text. The categories of appraisals about the design process, designed product, and designer are then applied toward the analysis of blogs, an online form of a designer's journal.

2. Affect in Design Thinking

In design cognition research, there is a tendency to circumscribe design as limited to the part of the life-cycle of the design process when a designer is producing specifications for a designed work, what others have called the production of knowledge representations (Visser 2004). This paper keeps open and flexible what is meant by designing as the set of activities associated with the lifecycle of artefact development, from planning to 'designing' to implementation to recycling.

Within these activities, feelings influence design in numerous ways. Take for example the human information processing strategy of rational decision-

making. In design, decision-making takes on the function of justifying the act of progressing from one stage of design to another or accepting a prototype or taking a decision to "come up with" another idea. Decision-making is a primary activity but is probably more far-reaching than just decisions about function, structure or behaviours. The full scope of interplay between affect and design cognition is beyond the purview of this paper. However, in order to study how affect affects thinking in design, examining the linguistic structure by which designers express affect can expose the way that affect is drawn into the way that designers perceive and interpret situations. Once this is isolated, we are then in the position to evaluate the effect of affect on their cognitive behaviours such as decision-making.

Thus, the question that is raised in the development of a linguistic framework to describe the way that designers represent affective states is to categorize the "situations" that may produce affective stimuli. We propose a hierarchical categorization in which, at the highest level, situations will involve either the process, product, or people; that is, designers will engage in reasoning about the process, product, or people. Other researchers have differentiated these categories as process-oriented or content-oriented (Stempfle and Badke-Schaub 2003). In this categorisation, process-oriented thinking includes reasoning about activity and events. Activity refers to discipline-specific tasks and the collective process of teamwork (Valkenburg 1998). People-oriented appraisal may refer to cognition or meta-cognitive thinking such as reflection, that is, thinking about thinking, and physical capabilities. Product-oriented thinking refers to reasoning about the goal space and the solution space of the designed artefact, about the function, form, behaviour, and meaning of the artefact.

This categorisation is also similar in concept to the OCC model of emotions (Ortony et al. 1988) and the object-relations theory of emotions. The OCC model proposes that emotions arise as a consequence of valenced reactions to objects, events, and agents. Applying this idea to the framework depicted in Figure 1, objects are the designed work (product), events are design processes (actions, activities and states of affairs that happen during design), and agents are people.

The object-relations theorists view emotions as value judgments ascribed to objects and persons outside of a person's own control and which are of importance for the person's flourishing (Nussbaum 2001). That is, an emotion is always intentionally directed at an object. The object-relations theorists construe the term object broadly to include agents, things, and events. In this framework for appraisal in design, an appraisal, as an emotion, is directed toward the product, process or a person where the notion of importance for flourishing is related to the designer's aim to produce a completed designed work. Given the correspondence between these models,

it may become possible to conjecture the emotional state of the designer given a statement of appraisal.

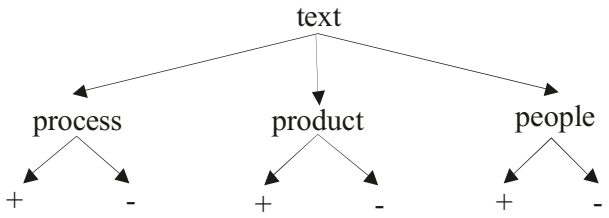


Figure 1. Categories of appraisal.

3. Linguistic Framework for Appraisal in Design

This paper applies the linguistic analysis technique of systemic functional linguistics (SFL) to analyze the set of resources applied by designers in the appraisal of design practice. The field of systemic functional linguistics examines how the semantic resources of a language create a system of options for speakers to enact the three main meta-functions of language: (1) ideational: to represent ideas; (2) interpersonal: to function as a medium of exchange between people; and (3) textual: to organize, structure, and hold itself together (Halliday 2004). Appraisal theory (Martin 2000) categorizes the system of linguistic options available for speakers to appraise as attitudinal positioning. Within this category are three sub-types of attitudinal positioning: 1) affect – how the speaker is emotionally disposed to the subject; 2) judgment – how the speaker assesses the subject with reference to behavioural norms and conventions; and 3) appreciation – how the speaker assesses the form, aesthetics, and appearance of objects, including other people. That is, any noun, verb, adjective, or adverb which functions to express meaning related to affect, judgment or appreciation is considered a term of appraisal. The clause within which the term appears is an appraisal.

For the analysis of design text, the linguistic framework of appraisal is modified and refined in several ways. First, the framework is modified to account for appraisal as a set of resources for attitudinal positioning and "reflection-in-action" (Schön 1983) in design in order to inquire and interpret ideas and activities. If emotions are evaluative appraisals, then language is most revealing as a source of difference in expressing emotions in design when we distinguish between linguistic resources for appraising processes (often called states of affairs and happenings in linguistics), product, and people. It therefore becomes important to know in the first instant to what external object this appraisal is directed toward. Second, the framework is modified to account for the idea that emotions necessarily

involve judgments about things which are of value to the designer whereas Martin's notion of judgment is limited to agents only. In design, judgments of objects and activities are routine; they are based on norms such as schools of design and accepted practices. Objects are judged in design and not just subjectively appreciated as Martin's framework prescribes. I further clarify that affect ought to distinguish between affect, cognitive and cognitive-behavioural dispositions in line with Ortony's affective lexicon. Finally, I move appraisal of a person's physical capabilities (as they relate to design) out of judgment and explicitly into a category called 'capability.' This category deals with appraisals of skill-based competencies and 'designerly' activities (Cross 1999), both of which are an integral aspect in appraising designers. In doing so, the notion of judgment and appreciation is also clarified: whereas external norms and standards are the basis for judgments, appreciation is based on interpersonal subjectivity. This clarification has a computational benefit in that knowing whether the appraisal is judgment or appreciation then differentiates the 'knowledge base' (as described in the OCC model of emotions) from which the appraisal is made: external (judgment) or internal (appreciation). It also affirms the object-relations differentiation between social, normative construction of judgment (external) and background 'life' experiences as the basis for judgment (internal).

Thus, I refine Martin's attitudinal positioning in the following ways. At the top level are a set of linguistic resources to accomplish reflection-in-action and attitudinal positioning toward product, process and people. Within this exist sub-types, again slightly modified in definition to accord with our understanding of design: 1) affect – how the designer describes affective, cognitive and cognitive-behavioural conditions that represent how the designer is thinking as well as how the designer is behaving in a cognitive-behavioural sense (Ortony et al. 1987); 2) judgment – how the designer appraises in relation to the accepted norms such as standards, industry best practices or normative design methods, and objective criteria established by the design brief; 3) appreciation – how the designer appraises in relation to personal experience (e.g., expertise and intuition) and subjective interpretations; and 4) capability – how the designer describes capability to or functioning of a person doing a design-related activity. This framework is summarized in Figure 2 and further clarified in the following sections.

In relation to Martin's treatment of engagement (e.g., "*I really do think* this is a good design.") and graduation (e.g., "This is a *fantastic* design!") in appraisals, I am in accord. But, these are finer linguistic details which this paper omits.

3.1. APPRAISAL OF PROCESS

It is generally regarded in design management that there exists four key aspects of successful design (or product development) in industry: 1) Strategy; 2) Organization; 3) Processes; and 4) Tools. Regardless, the appraisal of strategy, organization, processes and tools valuates either their capability to support, facilitate, or perform 'designing' or their current state of affairs. The appraisal of process is identified by taking stances towards tangible tasks and actions. If the appraisal makes references to established norms in design practice, then the appraisal is considered as judgment. Conversely, if the appraisal relies on experience and personal interpretations of the situation, it is considered a subjective appreciation.

In all of the process-oriented appraisal clauses, a tangible action is being evaluated, not the agent undertaking the action. The evaluation associates a position toward the state of being of the action. Appraisals of processes can normally be identified by asking, "What is being/was done?" and then "What is/was the stance toward the action?" The following questions guide the identification of appraisals towards design activities. For all of the examples which follow, italicized words are used to indicate distinguishing semantic and grammatical forms. Bold face terms indicate the appraisal word or words if there exists an intensifier. Because these expressions are taken from the design student blogs to be described in Section 4, they may contain spelling and some minor grammatical errors. The grammatical errors normally do not detract from the analysis unless they are significant enough such that the meaning of the clause is not evident.

Is the appraisal taken toward a specific task or action?

1. *Added a key* and *that should* ***help*** – Process (Positive Appreciation)

2. *I don't* ***regret*** *changing them* – Process/Task (Positive Appreciation)

3. *It certainly takes an* ***awfully long*** *time* to do stuff – Process (Negative Appreciation)

In the above example, the clause "to do stuff" is a continuation of the pronoun "It". The clause could have also been written as, "Doing stuff takes an awfully long time."

Is the appraisal taken towards the design process in general?

4. *Designing things is* ***painful***. – Process (Negative Appreciation)

5. *This is not the design process* at its ***finest***, – Process (Negative Judgment)

3.2. APPRAISAL OF PRODUCTS

Appraisals of products are one way in which designers offer subjective, personal assessments or apply normative judgments based on industry best practices, and external authoritative critique. Appraisals of products can justify (provide rationale) decisions taken during the design process. That is, their appraisals of products can explain how their *feelings* toward the designed object influenced the designing of the object. For example, a designer might feel that a current design concept is better than a prior concept (Love 1999) or better than available products on the market. In the appraisal of the product, the designer may rely on systems of linguistic resources that apply an external, normative *judgment* or a personal, subjective *appreciation*.

Whereas the appraisal of process is associated with a tangible action, an appraisal of a product is associated with an object. If this is based on formal analysis, we would consider this a judgment. If it is based solely on opinion, then it is a subjective appreciation. In the appraisals, the designers can either accept or resist critique by others (where the others include design norms set by authoritative persons and organizations within the field of design practice and not just the critique by other students and instructors in the class) which is categorized as *external valuation of products* or provide a personal appreciative stance on the designed object which is categorized as *subjective valuation of products*.

1. *The visualization can be* ***appreciated*** – Product (Positive External Judgment)

By writing the above valuation in the passive voice, the attitudinal positioning appeals to external valuation of aesthetics and behaviour based on the conventions of professionalism and cognitive ergonomics. In the next example, by attributing a statement to another person ('He said') the attitudinal positioning is also placed externally.

2. *He said* that *there is* ***too much*** *user interactivity* and that *he doesn't* ***like*** *the fact* that the time disappears when the animation is over. – Product (Negative External Judgment)

3. *My visualisation feels* ***so inadequate***. – Product (Negative Subjective Appreciation)

4. *I really* ***dislike*** *any design* that *chops and changes* between styles, layouts, formats, etc. – Product (Negative Appreciation)

5. *It just looks* ***unprofessional***, and *is* ***confusing*** for people. – Product (Negative External Judgment)

For the above clause, she makes an external valuation of meaning by deploying semantics which have official meanings. In the student's design discipline, professionalism relates to rules and heuristics for information design and interaction design. The reference to confusion pertains to cognitive ergonomics. Other disciplines may impose their own external valuations. In mechanical engineering, for example, these external valuations might relate to DFx rules (e.g., design for manufacturability, design for assembly).

Grammatically, external and internal valuations can be structurally differentiated. We have already noted the case of the passive voice in Clause 1. Also, the expression 'It just looks unprofessional' (as expressed by the student) is grammatically different from 'I think that it just looks unprofessional'. The grammatical form 'I think' indicates a personal mental state (opinion) rather than an extrinsic, attributive relationship between 'It' and 'unprofessional.' In other instances, domain knowledge is required to establish who or what entity is appraising. If the who or what is the designer, then the appraisal is subjective (appreciation). If the who or what is a third party or non-human agent (such as a design rule), then the appraisal is a judgment. Such a distinction is necessary to locate how subjectivity and authority operate within a design context.

Further, the appraisals may examine distinct aspects of a product: form (the structure of the object), functionality (what the object does), behaviour (how the object enacts the function), and the meaning of an object. Gero refers to the first three categories as the Function – Structure – Behaviour schema for a designed object (Gero 1990). Valuations of meaning are what Rothery and Stenglin (1999) named "social value." These values may be extrinsically held or intrinsically held. If the value is intrinsically held, then the valuation is subjective regarding the meaning of the product to the designer. Thus, a more complete system of appraisal of products should categorize by distinct aspects of a product, and then whether the appraisal is based on judgment (external valuation) or appreciation (subjective valuation). The following examples in Table 1 illustrates appraisals categorized by aesthetic, function, behaviour and meaning and then external valuation or subjective valuation. The + or – in each cell indicates if it is a positive or negative appraisal, respectively.

3.3. APPRAISAL OF PEOPLE

In the appraisal of people, the designer expresses subjective valuations of himself/herself or others. To limit the potential scope of appraisals of people, the taxonomy considers appraisals of a person's cognitive and physical states and capabilities. In line with the OCC model and the objection-relations theorists, appraisal of personal states includes internal, mental states of

affect, cognitive states, cognitive-behavioural states, and physical functioning or capability (the functioning of or capability to undertake a tangible design-oriented action). In distinguishing between cognitive and cognitive-behavioural, we follow the prescription by Ortony's affective lexicon. The affective lexicon distinguishes these three categories based on the "significant referential focus" of a word. Affective words express internal, mental states of being which do not have a significant cognitive or behavioural focus. Words with a cognitive focus "refer to aspects of knowing, believing or thinking. Specifically, they refer to such things as readiness, success, and desire." Words with a cognitive-behavioural focus refer to how a person is "thinking about a situation as well as to how one is acting." (Ortony et al. 1987).

TABLE 1. System of Appraisal of Products.

Category	External Valuation	Subjective Valuation
Form	*It* just *looks* ***unprofessional***, and is confusing for people. (–)	*I* ***like*** *the look* of the system I'm getting from my basic Flash skills. (+) *The icons look* ***very nice*** though. (+)
Function	*General layout did not facilitate* ***ease of use***. (–)	*I* also *think my solar system is providing a* ***good feel*** for the distance between people/planets. (+)
Behaviour	*It* just looks unprofessional, and *is* ***confusing*** for people. (–)	*I'm thinking* of *animating the blobs*, kind of like ripples on the surface of the water, of course *that* would be for pretty purposes only, which *wouldn't be* ***very effective***. (–)
Meaning	*The visualization can be* ***appreciated***. (+)	*I* ***like*** *that idea*. (+) *My thing is so* ***crappy*** so far I'm guna start again. (–)

In building up an outline of the type of appraisals of people a designer may conduct, we considered the following factors, each of which have an associated system of semantic resources:

Is the appraisal directed at another person and how the person is thinking and acting in a situation?

1. *You're doing* ***fine***. – People/Cognitive-Behavioural (Positive Appreciation)

Is the appraisal taken towards a self-assessment of capability?

2. *I'm* getting ideas, but ***not very good*** at "implementation." – People/Capability (Negative Appreciation)

3. Also, *I am trying to design* ***consistently***. – People/Capability (Positive Judgment)

4. *I* ***don't think*** *I* ***would be*** *able to make that* look real ***even*** *if I tried*. – People/Capability (Negative Appreciation)

Is the feeling directed towards a temporal cognitive state and frame of mind? For example, how does the designer feel at the point of starting the process of conceptualization (Solovyova 2003)?

5. *I am* ***confident*** – People/Cognitive (Positive Appreciation)

6. *I was* ***so in the zone*** – People/Cognitive (Positive Appreciation)

Is the feeling construed or directed towards a long-term, ongoing feeling? For example, does the designer feel if the work to date is sufficient to stop (sense of closure) (Love 1999)?

7. *I* still ***feel*** that *I'm* ***not completely*** *understanding* the visualization concept – People/Cognitive (Negative Appreciation)

Does the feeling relate to a future cognitive state? How does the designer feel about a potential outcome of a design move (Langan-Fox and Shirley 2003)?

8. *This option puts me* ***ill-at-ease*** if I pursue it – People/Affect (Negative Appreciation)

9. *I would be* ***at-ease*** if I purse this concept – People/Affect (Positive Appreciation)

To summarize, the structure of the language of appraisal in design can be characterized in the diagram of Figure 2.

The grammar of the language of appraisal varies at each level. At the first level, the taxonomy prescribes the intentional object towards which the appraisal is directed. The second level relates the characteristic of the object that is being appraised. The next level indicates the sub-type of the appraisal, whether it is related to affect, judgment or appreciation. If it is related to affect, the taxonomy further sub-divides the appraisal into affect and (physical, design-oriented) capability. The actual appraisal 'happens' at the far-right of the diagram as indicated by the + and –.

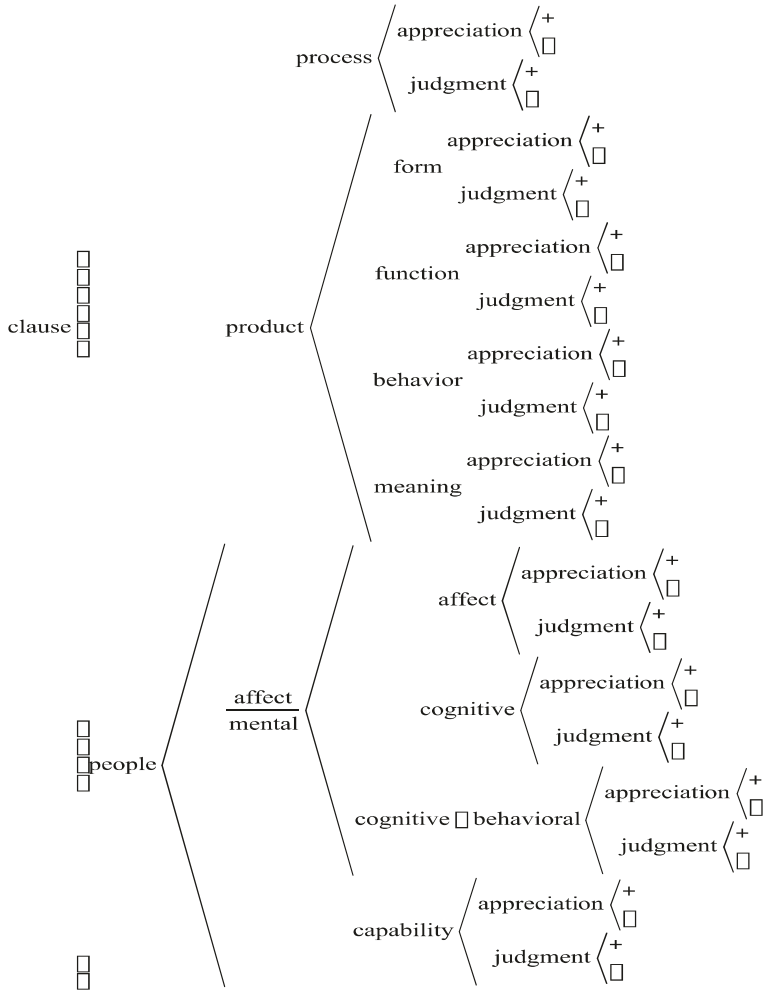


Figure 2. Structure of the language of appraisal in design.

4. Analysis of Blogs

4.1. LINGUISTIC ANALYSIS METHOD

The following section presents a qualitative analysis of three blogs by design students in a first year, first semester class relating to digital image design. This analysis is intended to offer an account of how design students deploy appraisal for reflection-in-action and attitudinal positioning. Here, our interest is not to describe how emotions affect design, but rather to illustrate how the taxonomy of the language of appraisal in design can be applied systematically to studying design, in this case, design as it happens in a studio learning environment.

The students blogged about their design process and their respective designed works (an information visualisation of an abstract phenomenon) in order to demonstrate learning about the practice of design. The three blogs chosen for analysis were taken from students who scored a mark of Distinction (>=75/100) or High Distinction (>=85/100). The basic reason for the choice of blogs is that these students wrote the most content in their blogs. Therefore, the choice of these blogs eliminates the erroneous interpretation that forms of appraisals may correlate with better design outcomes. This qualitative analysis is concerned with interpreting what the students attempt to achieve through appraisal in their blogs.

The technique of systemic functional linguistics attempts to eliminate subjectivity in linguistic analysis by following a prescriptive, objective method for the functional-semantic analysis of the grammar and the participants in the grammar. It overcomes the inter-coder reliability issue as only a single correct (with respect to the context of the text and the lexico-grammatical framework of Figure 2) analysis of the grammatical form of a sentence exists. While a full explanation of the SFL functional grammar analysis is beyond the scope of this paper (see Eggins 2004, pp. 206-253, for a detailed explanation), we provide a flavour for the analysis below in order to highlight the relatively high level of objectivity of the grammatical analysis. While necessary details about dealing with complex clauses, words that signify the process types and participants, semantic differences between form and behaviour, or affect and cognitive-behavioural and handling appreciation versus judgment are omitted, the process steps detail the formal analysis technique undertaken to analyze the semantic-grammatical form of the blogs.

The examples will use the clauses “It certainly takes an awfully long time to do stuff,” “It just looks unprofessional,” and “I don’t regret changing them” to illustrate the analysis. The analysis proceeds as follows:

1. Identify the verb clause. This is known as the Process.
2. Identify the Participants with the verb clause.

It	certainly	takes	an awfully long time	to do stuff
Participant		Process	Participant	

It	just	looks	painful
Participant		Process	Participant

I	don't	regret	changing them
Participant		Process	Participant

3. Using the rules of the TRANSITIVITY system in SFL, decide the appropriate process type: mental (thinking), material (doing), relational (having, being), existential (existing) or behavioural (behaving) and the corresponding participant types. This analysis does not include the verbal process type as it is not significant in design text. For the purposes of this analysis, we consider relational and existential as equivalent. Below are the clauses from above analyzed using the TRANSITIVITY system.

It	certainly	takes	an awfully long time	to do stuff
Actor	Intensifier	Process: material	Range	Actor

It	just	looks	painful
Carrier	Intensifier	Process: relational	Attribute

I	don't	regret	changing them
Senser		Process: mental	Phenomenon: act

4. As only a conscious agent may have a mental or behavioural process type, it is required to differentiate between the grammatical forms of statements such as "I like this design" (appraisal of a product) and "I feel disappointed" (appraisal of a person). As such, the second participant, called the non-active participant or Phenomenon, is used to distinguish the category.
    a. If the Phenomenon is an act, then the category is PROCESS. Proceed to Step 7.

   b. If the Phenomenon is an affect or intrinsic capability, then the category is PEOPLE. Proceed to Step 7.
   c. If the Phenomenon is an object, then the category is PRODUCT. Proceed to Step 7.
5. If the clause is material, then we must examine the semantic category of the Actor and the Goal/Range.
   a. If the Goal/Range is an agent, the clause belongs in the category PEOPLE. Proceed to Step 7.
   b. If the Actor or Goal/Range is an action or mental state, the clause belongs in the category PROCESS. Proceed to Step 7.
   c. If the Actor or Goal/Range is an object, the clause belongs in the category PRODUCT. Proceed to Step 7.
6. If the clause is relational or existential:
   a. If the Carrier and Attribute are a person, body, feeling, or cognition, then the clause belongs in the category PEOPLE. Proceed to Step 7.
   b. If the Carrier and Attribute are an act, event, phenomenon, or state, then the clause belongs in the category PROCESS. Proceed to Step 7.
   c. If the Carrier and Attribute are an object or substance, then the clause belongs in the category PRODUCT. Proceed to Step 7.
7. Given the top-level category, distinguish the sub-category using the semantic definitions of the Participants and Process types. Refer to the lexical categories of concepts in WordNet where possible. Some level of domain knowledge may be required, such as to distinguish between the function and the behaviour of an actor in a material process.
8. Given the top-level and sub-category, distinguish between appreciation and judgment for process and product appraisals, and affect or capability for people appraisals. For affective appraisals of people, further distinguish between affect, cognitive, or cognitive-behavioural appraisal following the affective lexicon prescribed by Ortony et al. (1987). In practice, one can produce labelled training data to train a machine learning algorithm for the text categorization operations of Steps 7 and 8.
9. Identify whether the sentiment expressed by the participant is positive or negative by locating the appraisal term(s). If there is no sentiment, the clause is not an appraisal. The appraisal terms are enclosed in {} in the following tables. Intensifiers are indicated by []. All three are negative appraisals.

It	[certainly]	takes	an {[awfully] long} time	to do stuff
Actor	Intensifier	Process: material	Range	Actor

It	[just]	looks	{painful}
Carrier	Intensifier	Process: relational	Attribute

I	don't	{regret}	changing them
Senser		Process: mental	Phenomenon: act

Based on the above procedure, three student blogs were analyzed. It should be noted first that the students wrote relatively fewer appraisals than descriptions of their designed work and in comparison to the total content of the blogs. It is for this reason that a quantitative analysis is not insightful. The relative dearth of appraisals may be due to the technical nature of the class and the instructor. That is, the socio-cultural code of the class may suppress appraisal by promulgating formal analysis as the currency of achievement in design.

The linguistic analysis of appraisal is usually quite complicated due to the complex clauses found in the text and the semantic interpretation of metaphors of appraisal rather than straightforward adjective and adverbial modifiers of nouns and verbs to express sentiment and polarity. The following is one such complex example.

> *I'm* going through my design and doing some last minute polishing, and *desperately trying to make my documentation* slightly less nonsensical than *what* i thought *was a literary masterpiece* at 3am.

This single sentence contains an appraisal of process and product, but the positive appraisal of the product is based on a metaphor – and one might question whether the student would still appraise the documentation as a 'literary masterpiece.'

4.2. RESULTS OF ANALYSIS

Based on the three blogs analysed, the appraisals accomplish three aims: rationalise decisions (justify design directions); generate kinship (garner social support for a design concept); and muse (maintain a space of possibilities).

4.2.1. Rationale

Appraisal of the product is the main way in which the students offered analysis and justification for design decisions. In the following example, the student appraises the use of text in her work:

> I've thought more about it and *i* do *think* that *text is* ***wrong*** in my visualisation. *i* can't help but *feel it is* ***a cop out***. Having said that, *i* also *think* that because of the ***abstract*** *nature of these visualisations*, *complete understanding is* ***impossible*** without text. so i am resolving to keep text and image separate throughout the whole project where possible. *i* ***like*** *that idea*

She begins by offering a negative appraisal of text, followed by the appraisal of the act of including text in her visualization as being a 'cop out'. As a consequence, she keys in the consequence of her appraisal 'to keep text and image separate throughout the whole project where possible' through the conjunction 'So' and the cognitive-behavioural state of 'resolving.' Finally, she concludes with a positive appraisal of the product, thereby justifying her design direction.

4.2.2. Kinship

In this design class, other than using the blogs to record their design activities and to exchange ideas and opinions, the students often used the blogs as a community forum for social support. In the following excerpt, a student expresses her negative affective state after a critique session.

> *I'm* ***depressed***!! We've just finished presenting our dynamic visualisations and I got *a few comments* that *I didn't* ***like***! I might be saying this because I spent A LOT of time trying to make this rotation thing work and *I was very* ***happy*** *that I made it work*! I know I shouldn't feel bad when people give me negative comments about my work. But that's what I feel right now! However, *I* kind of *feel* ***good*** that my visualisation had less comments than some other ones presented.

In response, another student wrote:

> Really, *I think comments are* ***just*** *opinions*. So its up to you to judge which you will take or reject.

The responding student downgrades comments by relating them to subjective opinions. Through this response, it is also evident that the preferred code of practice in the design studio is analysis rather than opinions – that is, facts derived from quantitative analysis rather than judgment.

By getting the students to engage in a conversation with each other electronically, and through those online conversations, the students became attuned to each other's emotional states. They offered support as a type of reciprocity for reading each other's blogs.

Muse

The final purpose for appraisals is to think while writing. In these instances, the appraisals offer the readers a glimpse into the mental state of the blogger. In the following excerpt, the student ruminates about her next design 'move.' While it is not clear what she will actually do, and we are told a bit about her intention to make the visualization 'look cuter', mostly, the contradictory appraisals of her affective state ('I am satisfied' 'I am not [satisfied]') and negative opinion of the current state of her designed work ('the animation … was really retarded') informs the reader that she is currently fuddling through the design process.

> I have spent all afternoon cleaning up my viz- the *animation* for April and May *was* ***really retarded*** in my last prototype... I also finished adding the remaining May data. AND I have just used graphics to label April and May- last time *I was lazy* and just dragged in the buttons I made for another prototype.. hehe. And I made an intro!!! Yay! *Nothing too* ***flash***, but *I was* ***reeeally fussy*** with it. But *i am satisfied*... Actually no, *I am not*. *I want it to look* ***cuter***. I may change it at the last minute if I have the time. Hopefully I can get a working timeline and mouse overs up and running tonight! For now, it's a shower and then dinner! =)

5. Conclusions

This paper presented a formal taxonomy and method of analysis that enables the identification of the grammatical and semantic structure of appraisal in design texts. Objectivity is introduced into the analysis by specifying a means to grammatically parse a clause, identify the process and participants, and then to categorise the clauses. The analysis does not account for textual marks that may signal an appraisal such as when a designer types the expression 'A LOT' or 'reeally.' Explicatives are also not included in the analysis. Based on the analytical framework, an analysis of appraisal in design student blogs was presented. The affective keys that the design students expressed were a basis for decision-making in design, generating kinship, and musing.

There is a growing body of research studying emotion in human information processing. The question that is posed and partially addressed in this research is how affective information is conveyed in designers' text and what functions the affective expressions serve. I suggest that the linguistic process of appraisal provides access to one dimension of affect. Clearly, there are other means of expressing emotions such as through body language, and physiology. The advantage of using linguistic expressions to study emotions is the unobtrusive monitoring that is necessary to record emotional data. The clear disadvantage is that not all emotions are accessible

by linguistic propositions. However, the cognitive/evaluative notion of emotion suggests a role for the use of the language of appraisal to get at the source of designer's emotions.

The principal advantage of the formal analysis of language reported in this paper is the ability to make the analysis computable. We are currently taking the results of grammatical analysis reported herein along with raters' evaluations of sentiments of design text to create training and validation data for a machine learning algorithm based on support vector machines. The algorithm will learn the semantic and grammatical features of the structural forms of appraisals. We plan to cross-validate the framework by analyzing a large body of design blogs by computer and interviewing the designers (bloggers) to ascertain whether patterns of appraisals in design text correspond to the designers' emotional states as would be suggested by the OCC model. With such information, we hope to be in the position to begin to answer how emotions influence 'rational' cognitive processes in design through empirical research.

Acknowledgements

This research is supported by an Australian Research Council grant DP0557346.

References

Cross, N: 1999, Natural intelligence in design, *Design Studies* **20**(1): 25-39.

Eggins, S: 2004, *An Introduction to Systemic Functional Linguistics*, Continuum International Publishing Group, London.

Ekman, P and Friesen, WV: 1975, *Unmasking the Face: A Guide to Recognizing Emotions from Facial Clues*, Prentice-Hall, Englewood Cliffs, NJ.

Ekman, P, Friesen, WV, O'Sullivan, M, Chan, A, Diacoyanni-Tarlatzis, I, Heider, K, Krause, R, LeCompte, WA, Pitcairn, T: 1987, Universals and cultural differences in the judgments of facial expressions of emotion, *Journal of Personality and Social Psychology* **53**(4): 712-717.

Fiedler, K: 2000, Toward an integrative account of affect and cognition phenomena using the BIAS computer algorithm, *in* JP Forgas (ed) *Feeling and Thinking: The Role of Affect in Social Cognition*, Maison des Sciences de l'Homme and Cambridge University Press, Cambridge, pp. 223-252.

Forgas, JP: 2000, Feeling and thinking: Summary and integration, *in* JP Forgas (ed) *Feeling and Thinking: The Role of Affect in Social Cognition*, Maison des Sciences de l'Homme and Cambridge University Press, Cambridge, pp. 387-406.

Gero, JS: 1990, Design prototypes: A knowledge representation schema for design, *AI Magazine* **11**(4): 26-36.

Halliday, MAK: 2004, *An Introduction to Functional Grammar*, Arnold, London.

Langan-Fox, J and Shirley, DA: 2003, The nature and measurement of intuition: Cognitive and behavioral interests, personality, and experiences, *Creativity Research Journal* **15**(2-3): 207-222.

Love, T: 1999, Computerising affective design computing, *International Journal of Design Computing* **2**: Available Online: http://www.arch.usyd.edu.au/kcdc/journal/vol2/love/cadcmain.htm.

Love, T: 2001, Concepts and affects in computational and cognitive models of designing, *in* JS Gero and ML Maher (eds), *Computational and Cognitive Models of Creative Design*, University of Sydney, Sydney, pp. 3-23.

Martin, JR: 2000, Beyond exchange: APPRAISAL systems in English, *in* S Hunston and G Thompson (eds), *Evaluation in Text: Authorial Stance and the Construction of Discourse*, Oxford University Press, Oxford, pp. 142-175.

Martin, JR and Rose, D: 2003, Appraisal: Negotiating attitudes, *in* JR Martin and D Rose (eds), *Working with discourse: meaning beyond the clause*, Continuum, London, pp. 22-65.

Newell, A: 1990, *Unified Theories of Cognition*, Harvard University Press, Cambridge, MA.

Nussbaum, MC: 2001, *Upheavals of Thought: The Intelligence of Emotions*, Cambridge University Press, Cambridge.

Ortony, A, Clore, GL and Collins, A: 1988, *The Cognitive Structure of Emotions*, Cambridge University Press, Cambridge.

Ortony, A, Clore, GL and Foss, MA: 1987, The referential structure of the affective lexicon, *Cognitive Science* **11**(3): 341-364.

Pennebaker, JW and King, LA: 1999, Linguistic styles: Language use as an individual difference, *Journal of Personality and Social Psychology* **77**(6): 1296-1312.

Rothery, J and Stenglin, M: 1999, Interpreting literature: The role of appraisal, *in* L Unsworth (ed) *Researching Language in Schools and Communities: Functional Linguistic Perpectives*, Cassell Academic, London, pp. 222-244.

Schön, DA: 1983, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York.

Solovyova, I: 2003, Conjecture and emotion: An investigation of the relationship between design thinking and emotional content, *in* N Cross and E Edmonds (eds), *Expertise in Design: Design Thinking Research Symposium 6*, Creativity and Cognition Studios Press, Sydney, Available Online: http://research.it.uts.edu.au/creative/design/papers/24SolovyovaDTRS6.pdf, last accessed March 2006.

Stempfle, J and Badke-Schaub, P: 2003, Thinking in design teams - an analysis of team communication, *Design Studies* **22**: 473-496.

Turney, PD and Littman, ML: 2003, Measuring praise and criticism: Inference of semantic orientation from association, *ACM Transactions on Information Systems* **21**(4): 315-346.

Valkenburg, RC: 1998, Shared understanding as a condition for team design, *Automation in Construction* **7**(2-3): 111-121.

Visser, W: 2004, *Dynamic Aspects of Design Cognition*, Institut National de Recherche en Informatique et en Automatique (INRIA), Paris.

FORMAL METHODS IN DESIGN

A function-behaviour-structure ontology of processes
John S Gero and Udo Kannengiesser

From form to function: From SBF to DSSBF
Patrick Yaner and Ashok Goel

Formal description of concept-synthesizing process for creative design
Yukari Nagai and Toshiharu Taura

Robustness in conceptual designing: Formal criteria
Kenneth Shelton and Tomasz Arciszewski

A FUNCTION-BEHAVIOUR-STRUCTURE ONTOLOGY OF PROCESSES

JOHN S GERO AND UDO KANNENGIESSER
University of Sydney, Australia

Abstract. This paper shows how the function-behaviour-structure (FBS) ontology can be used to represent processes despite its original focus on representing objects. The FBS ontology provides a uniform framework for classifying processes and includes higher-level semantics in their representation. We demonstrate that this ontology supports a situated view of processes based on a model of three interacting worlds.

1. Introduction

Ontologies are structured conceptualisations of a domain in terms of a set of entities in that domain and their relationships. They provide uniform frameworks to identify differences and similarities that would otherwise be obscured. In the design domain, a number of ontologies have been developed to represent objects, specifically artefacts. They form the basis for a common understanding and terminological agreement on all relevant properties of a specific artefact or class of artefacts. Ontologies can then be used to represent the evolving states of designing these artefacts or as knowledge representation schemas for systems that support designing.

Design research is a field that has traditionally shown particular interest in explicit representations of processes besides objects. A number of process taxonomies have been created that classify different design methods (e.g. Cross (1994), Hubka and Eder (1996)). However, most of this work has not been based on process ontologies, which makes comparison of the different taxonomies difficult. Some of the efforts towards stronger ontological foundations for process representation have been driven by the need to effectively plan and control design and construction processes. For example, recent work on 4D CAD systems links 3D object models to project schedules (Haymaker and Fischer 2001). Process ontologies used in the design field include Cyc (Lenat and Guha 1990), IDEF0 (NIST 1993) and PSL (NIST 2000).

J.S. Gero (ed.), Design Computing and Cognition '06, 407–422.
© 2006 *Springer. Printed in the Netherlands.*

Most process ontologies and representations have a view of processes that is based on activities and/or their pre- and post-conditions. Higher-level semantics are generally not included in most process ontologies. Such semantics are needed to guide the generation, analysis and evaluation of a variety of processes. As research increasingly focuses on automating parts of the selection or synthesis of processes, existing process ontologies provide inadequate representations for computational support.

An ontology that supports higher-level semantics is Gero's (1990) function-behaviour-structure (FBS) ontology. Its original focus was on representing artificial objects. In this paper we show how this focus can be extended to include processes. We then apply Gero and Kannengiesser's (2004) three-world model to develop a situated view of processes, which also demonstrates some of the benefits of including higher-level semantics into process representations.

2. The FBS Ontology

2.1. THE FBS VIEW OF OBJECTS

The FBS ontology provides three high-level categories for the properties of an object:

1. *Function* (F) of an object is defined as its teleology, i.e. "what the object is for".
2. *Behaviour* (B) of an object is defined as the attributes that are derived or expected to be derived from its structure (S), i.e. "what the object does".
3. *Structure* (S) of an object is defined as its components and their relationships, i.e. "what the object consists of". The structure (S) of most objects can be described in terms of geometry, topology and material.

Humans construct connections between F, B and S through experience and through the development of causal models based on interactions with the object. Specifically, function (F) is ascribed to behaviour (B) by establishing a teleological connection between the human's goals and observable or measurable effects of the object. Behaviour (B) is causally connected to structure (S), i.e. it can be derived from structure using physical laws or heuristics. There is no direct connection between function (F) and structure (S), which is known as the "no-function-in-structure" principle (De Kleer and Brown 1984).

The generality of the FBS ontology allows for multiple views of the same object. This enables the construction of different models depending on their purpose. For example, an architectural view of a building object includes different FBS properties than a structural engineering view. This is most

striking for the building's structure (S): Architects typically view this structure as a configuration of spaces, while engineers often prefer a disjoint view based on floors and walls.

Multiple views can also be constructed depending on the required level of aggregation. This allows modelling objects as assemblies composed of sub-assemblies and individual parts. Each of these components can again contain other sub-assemblies or parts. No matter which level of aggregation is required, the FBS ontology can always be applied.

2.2. THE FBS VIEW OF PROCESSES

Objects and processes have traditionally been regarded as two orthogonal views of the world. The difference between these views is primarily based on the different levels of abstraction involved in describing what makes up their structure. The structure of physical or virtual objects consists of representations of material, geometry and topology. These representations can be easily visualised and understood. Processes are more abstract constructs that include transitions from one state of affairs to another.

The well-established field of object-oriented software engineering has most explicitly demonstrated how abstraction can overcome the traditional division between the object-centred and the process-centred view of the world. Object-oriented software commonly uses a set of program elements that are conceived of as representing objects as well as processes that operate on them. All of these program elements encapsulate state variables and define methods to enable interactions with other elements.

The high-level categorisations provided by the FBS ontology can be used to create a similar, integrative view that treats objects and processes in a uniform manner. This is possible because the FBS ontology does not include the notion of time. While on an instance level this notion is fundamental to the common distinction between objects and processes, on an ontological level there is no time-based difference between them. All states of any entity at any point in time can be described by a set of properties that can be classified as function (F), behaviour (B) and structure (S).

It is not hard to see that the notion of function (F) applies to any entity as it only accounts for the observer's goals, independent of the entity's embodiment as an object or as a process.

Behaviour (B) relates to those attributes of an entity that allow comparison on a performance level rather than on a compositional level. Such performance attributes are representations of the effects of the entity's interactions with its environment. Typical behaviours (B) of processes are speed, rate of convergence, cost, amount of space required and accuracy.

While process function (F) and process behaviour (B) are not fundamentally different to object function and object behaviour, process

structure (S) is clearly distinctive. It includes three classes of components and two classes of relationships, Figure 1.

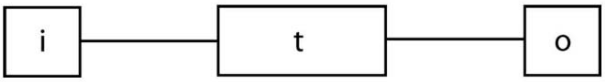


Figure 1. The structure (S) of a process. (i = input; t = transformation; o = output).

The components are

- an input (i),
- a transformation (t) and
- an output (o).

The relationships connect

- the input and the transformation (i – t) and
- the transformation and the output (t – o).

2.2.1. Input (i) and Output (o)

The input (i) and the output (o) structure elements represent properties of other entities in terms of their variables and/or their values. For example, the process of transportation changes only the values for the location of a (physical) object (e.g. the values of its x-, y- and z-coordinates). As the input (i) and output (o) contain the same variables here, such a process can be characterised as homogenous. Heterogenous processes, in contrast, use disparate variables as input (i) and output (o). For example, the process of electricity generation takes mechanical motion as input (i) and produces electrical energy as output (o).

Input (i) and output (o) may refer not only to (properties of) objects but also to (properties of) other processes. For example, it is not uncommon for software procedures to accept the output of other procedures as their input (i) or to return procedure calls as their output (o). All variables and values used as input (i) and output (o) of a process may refer to the function, behaviour or structure of other objects or processes.

2.2.2. Transformation (t)

A common way to describe the transformation (t) of a process is in terms of a plan, a set of rules or other procedural descriptions. A typical example is a software procedure that is expressed in source code or as a UML[1] activity diagram. Such descriptions are often viewed as a collection of subordinate processes. In the software example, this is most explicit when a procedure

[1] Unified Modeling Language

calls other procedures that are possibly located in other program components or other computers. Every sub-process can again be modelled in terms of function, behaviour and structure.

The transformation (t) of a process can also be described in terms of an object. Take the software example, the transformation (t) of a process may be viewed simply as the object (in the context of object-oriented programming) that provides appropriate methods to carry out that process. Another example for a transformation (t) can be a computational agent. Such object-centred descriptions of transformations (t) are often used when not much is known about the internal mechanisms of that transformation or when not much is gained by explicitly modelling these mechanisms. In some cases, the transformation (t) is only a "black box" that merely serves to connect the input (i) to the output (o). For example, the programmer designing the software procedure constructs an extensive set of properties related to the transformation (t). In contrast, for the users of that procedure the transformation (t) is often a "black box", as the source code is usually not available or relevant. They base their views of the process structure (S) mainly on the input (i) and output (i) variables that are specified in the application programming interface (API).

2.2.3. Relationships

The relationships between the three components of a process are usually uni-directional from the input (i) to the transformation (t) and from the transformation (t) to the output (o). For iterative processes the t – o relationship is bi-directional to represent the feedback loop between the output (o) and the transformation (t).

2.2.4. Some Process Classifications Based on the FBS Ontology

The FBS view of processes provides a means to classify different instances of design processes according to differences in their function, behaviour or structure. Take Gero's (1990) eight fundamental classes of processes involved in designing, they can be distinguished by differences in their input (i) and output (o). For example, while synthesis is a transformation of expected behaviour (i) into structure (o), analysis transforms structure (i) into behaviour (o). Within each of these fundamental processes we can identify different instances if we reduce the level of abstraction at which input and output are specified. For example, different instances of the process class analysis can be defined based on the specific kind of output they produce: stress analysis computes stress (o), thermal analysis computes temperature (o), cost analysis computes cost (o), etc. Other process instances can be based on the transformation (t). For example, the synthesis of a design object can be carried out using a range of different transformations (t)

or techniques to map expected behaviour onto structure. Examples include case-based reasoning, genetic algorithms or gradient-based search methods.

While most process classifications and taxonomies are based on differences in structure (S), processes can also be distinguished according to their behaviour (B) and function (F). For example, design optimization processes can be characterised on the basis of differences in their speed, differences in the amount of space they require or other behaviours (B). Another example has been provided by Sim and Duffy (1998), who propose a multi-dimensional classification of machine learning processes in design that can partially be mapped on structure (S) and function (F) of a process. Specifically, learning processes are grouped according to input knowledge (i), knowledge transformers (t), output knowledge (o) and learning goal (F), among others.

3. Situated FBS Representations of Processes

3.1. SITUATEDNESS

Designing is an activity during which designers perform actions in order to change their environment. By observing and interpreting the results of their actions, they then decide on new actions to be executed on the environment. This means that the designers' concepts may change according to what they are "seeing", which itself is a function of what they have done. One may speak of an "interaction of making and seeing" (Schön and Wiggins 1992). This interaction between the designer and the environment strongly determines the course of designing. This idea is called situatedness, whose foundational concepts go back to the work of Dewey (1896) and Bartlett (1932).

In experimental studies of designers, phenomena related to the use of sketches, which support this idea, have been reported. Schön and Wiggins (1992) found that designers use their sketches not only as an external memory, but also as a means to reinterpret what they have drawn, thus leading the design in a new direction. Suwa et al. (1999) noted, in studying designers, a correlation of unexpected discoveries in sketches with the invention of new issues or requirements during the design process. They concluded that "sketches serve as a physical setting in which design thoughts are constructed on the fly in a situated way".

Gero and Fujii (2000) have developed a framework for situated cognition, which describes the designer's interpretation of their environment as interconnected sensation, perception and conception processes. Each of them consists of two parallel processes that interact with each other: A *push process* (or data-driven process), where the production of an internal representation is driven ("pushed") by the environment, and a *pull process*

(or expectation-driven process), where the interpretation is driven ("pulled") by some of the designer's current concepts, which has the effect that the interpreted environment is biased to match the current expectations.

The environment that is interpreted can be external or internal to the agent. The situated interpretation of the internal environment accounts for the notion of constructive memory. The relevance of this notion in the area of design research has been shown by Gero (1999). Constructive memory is best exemplified by a quote from Dewey via Clancey (1997): "Sequences of acts are composed such that subsequent experiences categorize and hence give meaning to what was experienced before". The implication of this is that memory is not laid down and fixed at the time of the original sensate experience but is a function of what comes later as well. Memories can therefore be viewed as being constructed in response to a specific demand, based on the original experience as well as the situation pertaining at the time of the demand for this memory. Therefore, everything that has happened since the original experience determines the result of memory construction. Each memory, after it has been constructed, is added to the existing knowledge (and becomes part of a new situation) and is now available to be used later, when new demands require the construction of further memories. These new memories can be viewed as new interpretations of the augmented knowledge.

The advantage of constructive memory is that the same external demand for a memory can potentially produce a different result, as newly acquired experiences may take part in the construction of that memory. Constructive memory can thus be seen as the capability to integrate new experiences by using them in constructing new memories. As a result, knowledge "wires itself up" based on the specific experiences it has had, rather than being fixed, and actions based on that knowledge can be altered in the light of new experiences.

Situated designing uses first-person knowledge grounded in the designer's interactions with their environment (Bickhard and Campbell 1996; Clancey 1997; Ziemke 1999; Smith and Gero 2005). This is in contrast to static approaches that attempt to encode all relevant design knowledge prior to its use. Evidence in support of first-person knowledge is provided by the fact that different designers are likely to produce different designs for the same set of requirements. And the same designer is likely to produce different designs at different points in time even though the same requirements are presented. This is a result of the designer acquiring new knowledge while interacting with their environment.

Gero and Kannengiesser (2004) have modelled situatedness as the interaction of three worlds, each of which can bring about changes in any of the other worlds. The three worlds include the observer's external world, interpreted world and expected world, Figure 2.

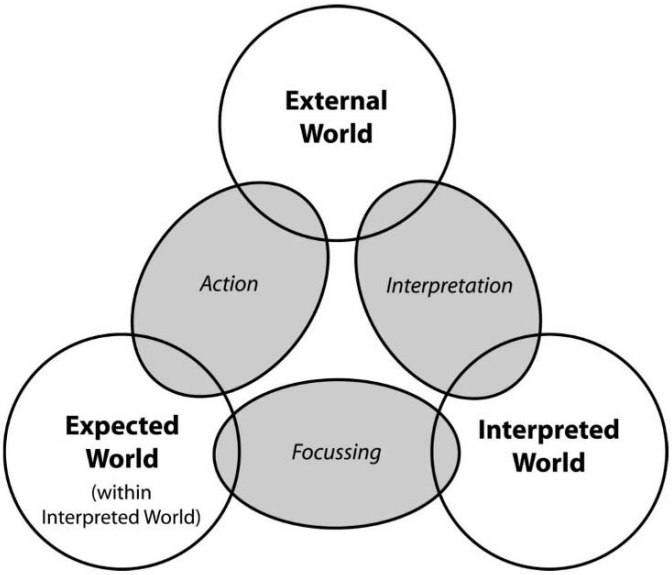


Figure 2. Situatedness as the interaction of three worlds.

The *external world* is the world that is composed of representations outside the observer (or designer).

The *interpreted world* is the world that is built up inside the designer in terms of sensory experiences, percepts and concepts. It is the internal representation of that part of the external world that the designer interacts with.

The *expected world* is the world imagined actions will produce. It is the environment in which the effects of actions are predicted according to current goals and interpretations of the current state of the world.

These three worlds are linked together by three classes of connections. *Interpretation* transforms variables which are sensed in the external world into the interpretations of sensory experiences, percepts and concepts that compose the interpreted world. *Focussing* takes some aspects of the interpreted world, uses them as goals in the expected world and suggests actions, which, if executed in the external world should produce states that reach the goals. *Action* is an effect which brings about a change in the external world according to the goals in the expected world.

3.2. CONSTRUCTING DIFFERENT VIEWS FOR DIFFERENT PURPOSES

Gero and Kannengiesser's (2004) three-world model can be used to construct a situated FBS view of processes. The main basis for creating situated representations is the distinction between the external and the

interpreted world. Locating function (F), behaviour (B) and structure (S) of a process in each of these worlds, Figure 3, results in six ontological categories:

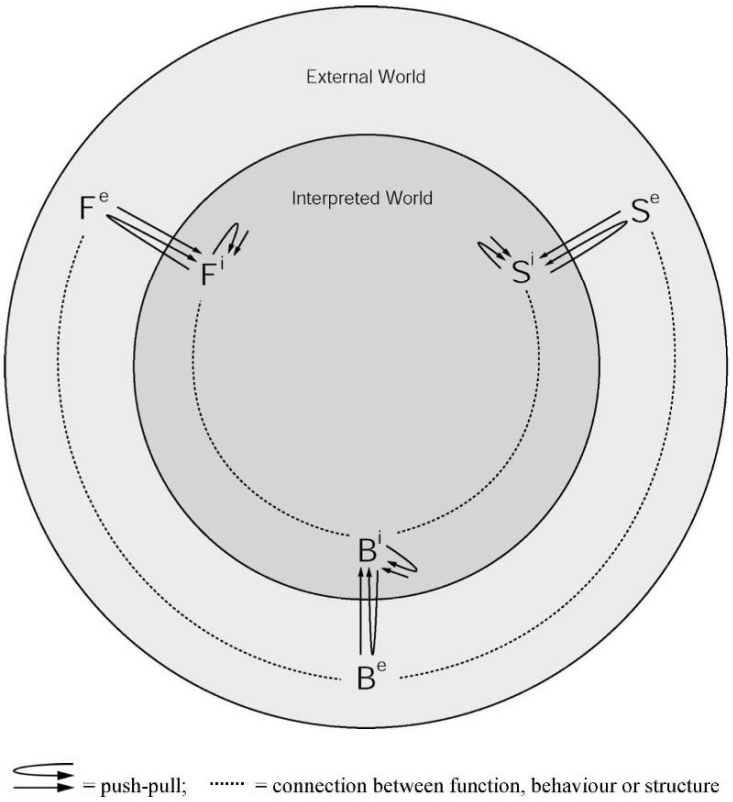


Figure 3. External and interpreted FBS representations of processes.

1. external function (F^e)
2. external behaviour (B^e)
3. external structure (S^e)
4. interpreted function (F^i)
5. interpreted behaviour (B^i)
6. interpreted structure (S^i)

Process representations of categories 4, 5 and 6 are generated via push-pull mechanisms involving only the internal world (constructive memory) or both internal and external worlds (interpretation).

3.2.1. External vs. Interpreted Structure of a Process

Most design ontologies cannot deal with different interpretations of a process, as they do not distinguish between external and interpreted worlds.

Such interpretations are often required for representing process structure (S). This is due to a number of reasons.

First, many instances of external process structure (S^e) are transient and time-based. Delineating the components of the process (i.e. input, transformation and output) from one another as well as from other entities in the external world then requires acts of discretisation from continuous flows of events according to the observer's current knowledge and goals. For example, it is possible to view the intermediate results of an iterative process as part of its transformation (t) or, alternatively, as part of its output (o).

Second, the kind of components of the process structure (S) and the level of detail used to describe them are similarly dependent on the stance of the observer. One example, already mentioned in Section 2.2.2, is the range of possible views of the transformation (t) from a detailed procedural plan to an object or a simple "black box". There are also many examples for disparate views of the input (i) and output (o) of the same process. Take a pressing process in the automotive industry: A manufacturing engineer generally views the input and the output of this process in terms of geometry of the sheet steel to be transformed. In contrast, a costing expert typically views the input and the output of the same process in terms of (material, labour, etc.) cost and yield, respectively. Similar view-dependent examples have been presented by NIST (2004).

3.2.2. External vs. Interpreted Behaviour of a Process

The distinction between external and interpreted worlds is also useful when dealing with the performance or behaviour (B) of a process. This allows different observers to reason about different performance aspects of a process according to the current situation. For example, the cost of burning fuel (available in the external world as external behaviour (B^e)) might be important for the owner of a car; however, this cost is usually not directly relevant for the hitchhiker sitting on their passenger seat. Another example is the amount of memory space needed by a particular computational process. This behaviour (B) is usually worth considering for users only if their hardware resources are limited for current purposes.

The kind of interpreted behaviour (B^i) that an observer is interested in also affects the way in which that observer interprets the structure (S) that is responsible for causing that behaviour. This is the case when no external behaviour (B^e) and no memories of previous interpreted behaviour (B^i) are available, and the interpreted behaviour (B^i) must be derived from structure. If, for instance, the speed of a process is to be measured, then a structural description of the input (i) and output (o) of that process must be produced that contains references to some quantities and time units. If the amount of space required by the process is to be measured, then there must be a

structural description that provides sufficient detail about the path of transformation (t) for given inputs (i) and outputs (o).

3.2.3. External vs. Interpreted Function of a Process

The need to separate the interpreted from the external world is most obvious for the function (F) of a process. Individual observers have the autonomy to interpret function according to their own goals and desires that are likely to differ from others. They may come up with various interpreted process functions (F^i), which may be independent of the constraints imposed by process structure and behaviour. For example, it is solely dependent on an observer's previous experience or current goals if they ascribe the function "operate time-efficiently" to a manufacturing process, even though the exact speed of that process (as its behaviour) may be given.

3.3. CONSTRUCTING DIFFERENT PURPOSES FROM DIFFERENT VIEWS

Let us add the expected world to the interpreted and external world, Figure 4. The number of ontological categories now increases to nine:

1. external function (F^e)
2. external behaviour (B^e)
3. external structure (S^e)
4. interpreted function (F^i)
5. interpreted behaviour (B^i)
6. interpreted structure (S^i)
7. expected function (Fe^i)
8. expected behaviour (Be^i)
9. expected structure (Se^i)

The distinction between the interpreted and the expected world reflects the potential gap between the perceived and the desired state of the world. Such a gap usually results in an action to change the external world according to the goals in the expected world.

3.3.1. External, Interpreted and Expected Structure of a Process

Representations of process structure (S) in the expected world describe the composition of desired processes. Actions can then be performed to realise (represent) the desired processes in the external world. One example of such processes is a strategy. One distinguishing feature of strategies is that the transformation (t) components of their structure (S) are viewed as actions or sequences of actions, undertaken either by individuals (Gruber 1989) or by organisations (Chandler 1962). These actions can then be interpreted again as part of an interpreted process structure (S^i) that may be different from the initial, expected process structure (Se^i). New strategies can be adopted by transferring interpreted process structures (S^i) into the expected world.

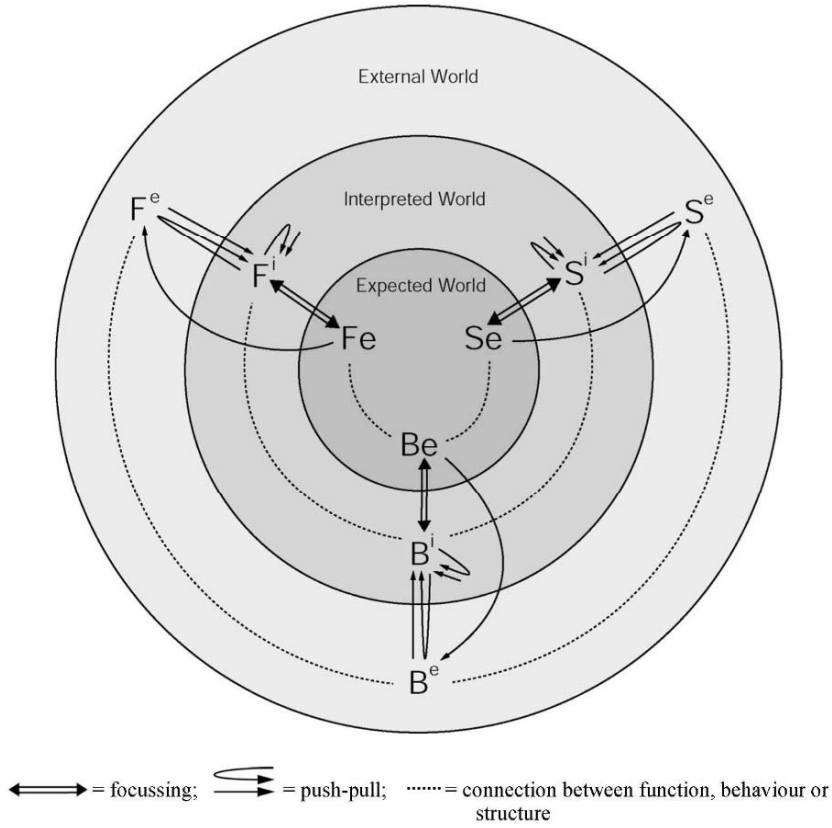


Figure 4. External, interpreted and expected FBS representations of processes.

The interaction between the external, interpreted and expected structure (S) of strategies is an instance of Schön's (1983) concept of "reflection-in-action". It allows for reflective reasoning about one's interactions with the external world, which has the potential of substantially changing current strategies (Hori 2000). Work in management science has established the term "strategizing" to denote the interactive construction of new strategies by cycles of interpretation and action (Cummings and Wilson 2003). Strategizing combines the traditional idea of top-down implementation of pre-formed strategies with more recent models of bottom-up recognition of new strategies as "patterns in a stream of actions" (Mintzberg and Waters 1985).

It has frequently been suggested that new strategies are recognised by identifying and eliminating redundant steps (Roberts and Newton 2001). This complies with the notion of emergence, which is a general mechanism for deriving new design concepts (Gero 1996). Emergence of design

strategies has been demonstrated by Nath and Gero (2004) to allow a computational system to acquire and reuse search (process) knowledge encoded as rules. The system can identify mappings between past design contexts and design decisions that led to useful results in these contexts. It then constructs new rules from these mappings using explanation-based learning.

Besides emergence, a number of other mechanisms may bring about new strategies. These are mutation, combination, analogy and first principles (Gero 1996). Not much research has been undertaken to date to apply them to strategies.

3.3.2. External, Interpreted and Expected Behaviour of a Process

Differences between the interpreted and the expected world at the level of the behaviour (B) of a process are, for instance, what project managers have to deal with. They represent gaps between the actual (interpreted) and the desired (expected) state of a process in terms of performance. Common examples include the speed, cost and accuracy of a process that may diverge from the corresponding target values specified in the project plan. There are two possibilities to reduce or eliminate the gap between the interpreted and the expected behaviour (Be^i) of the process. First, corrective action may be taken to change the current process performance in the external world (B^e) that would then change the corresponding interpreted performance (B^i). Second, the expected behaviour (Be^i) may be adjusted to the current state of the process in order to satisfice the project plan.

The performance or behaviour (B) level has also been used to justify the selection of a particular design strategy (Clibbon and Edmonds 1996). Chandrasekaran et al. (1993) have similarly included behaviour (B) into representations of design rationale to retrospectively document and explain decisions taken in a design process. The distinction between interpreted and expected process behaviour (B) allows comparing the performance of alternative strategies and ultimately selecting one of them.

3.3.3. External, Interpreted and Expected Function of a Process

The distinction between interpreted and expected function (F) of a process describes the gap between potentially adoptable and currently focussed purposes ascribed to the process. Similar to behaviour (B), this gap may be reduced or eliminated through action to modify external function (F^e) or through adoption of new expected function (Fe^i).

Representations of expected function (F) can also be used to provide constraints for selecting the behaviour (B) and structure (S) of processes via the connections between function, behaviour and structure. They link the performance and composition of processes to the current teleological context

by adding functional requirements. For example, von der Weth (1999) has suggested that expectations of functions (F) such as "carefulness" or "thoughtfulness" support the selection of strategies that are adapted to the degree of complexity, novelty and dynamism of a given situation.

4. Discussion

We have presented the FBS ontology as a structured conceptualisation of the domain of processes. We claim that any class of process can be represented using this ontology. A number of examples of processes in the design domain have been described earlier in this paper. Our ontology provides a uniform representation that allows locating as well as distinguishing between them.

Integrating function and behaviour in a process ontology adds higher-level semantics to process representations, which accounts for their applicability in a purposive context. This is useful for knowledge representations of processes, as they can be deployed by a knowledge-based system to select, compare and execute specific processes according to its current goals. Such knowledge representations are equivalent to Gero's (1990) design prototypes based on the FBS ontology for design objects. The ability to support different views and purposes of processes at functional, behavioural and structural levels increases flexibility and applicability of the system in different situations.

Another major advantage of the presented FBS ontology of processes is that it uses the same fundamental constructs – function, behaviour and structure – as for objects. This allows developing design systems or agents that can flexibly reason about a variety of objects and processes without having to implement different, specialised cognitive mechanisms. As everything in the world looks the same when viewed in terms of FBS, only one cognitive mechanism is required. Reflective, meta-cognitive systems (e.g. Singh et al. (2004)) would particularly benefit from our ontological approach to processes as it avoids implementing multiple layers of reasoning.

Acknowledgements

This research is supported by a grant from the Australian Research Council, grant DP0559885.

References

Bartlett, FC: 1932 reprinted in 1977, *Remembering: A Study in Experimental and Social Psychology*, Cambridge University Press, Cambridge.

Bickhard, MH and Campbell, RL: 1996, Topologies of learning, *New Ideas in Psychology* **14**(2): 111-156.

Chandler, AD: 1962, *Strategy and Structure*, MIT Press, Cambridge.

Chandrasekaran, B, Goel, AK and Iwasaki, Y: 1993, Functional representation as design rationale, *IEEE Computer* **26**(1): 48-56.

Clancey, WJ: 1997, *Situated Cognition: On Human Knowledge and Computer Representations*, Cambridge University Press, Cambridge.

Clibbon, K and Edmonds, E: 1996, Representing strategic design knowledge, *Engineering Applications of Artificial Intelligence* **9**(4): 349-357.

Cross, N: 1994, *Engineering Design Methods: Strategies for Product Design*, John Wiley & Sons, Chichester.

Cummings, S and Wilson, D (eds): 2003, *Images of Strategy*, Blackwell Publishers, Oxford.

De Kleer, J and Brown, JS: 1984, A qualitative physics based on confluences, *Artificial Intelligence* **24**: 7-83.

Dewey, J: 1896 reprinted in 1981, The reflex arc concept in psychology, *Psychological Review* **3**: 357-370.

Gero, JS: 1990, Design prototypes: A knowledge representation schema for design, *AI Magazine* **11**(4): 26-36.

Gero, JS: 1999, Constructive memory in design thinking, *in* G Goldschmidt and W Porter (eds), *Design Thinking Research Symposium: Design Representation,* MIT, Cambridge, MA, pp. 29-35.

Gero, JS and Fujii, H: 2000, A computational framework for concept formation for a situated design agent, *Knowledge-Based Systems* **13**(6): 361-368.

Gero, JS and Kannengiesser, U: 2004, The situated function-behaviour-structure framework, *Design Studies* **25**(4): 373-391.

Gruber, TR: 1989, Automated knowledge acquisition for strategic knowledge, *Machine Learning* **4**: 293-336.

Haymaker, J and Fischer, M: 2001, Challenges and benefits of 4D modeling on the Walt Disney concert hall project, *CIFE Working Paper #64*, Center for Integrated Facility Engineering, Stanford University, Stanford, CA.

Hori, K: 2000, An ontology of strategic knowledge: Key concepts and applications, *Knowledge-Based Systems* **13**: 369-374.

Hubka, V and Eder, WE: 1996, *Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge*, Springer-Verlag, Berlin.

Lenat, DB and Guha, RV: 1990, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley, Reading.

Mintzberg, H and Waters, JA: 1985, Of strategies, deliberate and emergent, *Strategic Management Journal* **6**(3): 257-272.

Nath, G and Gero, JS: 2004, Learning while designing, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **18**(4): 315-341.

NIST: 1993, Integration definition for function modeling (IDEF0), *Federal Information Processing Standards Publication 183*, National Institute of Standards and Technology, Gaithersburg, MD.

NIST: 2000, The process specification language (PSL): Overview and version 1.0 specification, *NIST Internal Report 6459*, National Institute of Standards and Technology, Gaithersburg, MD.

NIST: 2004, Inputs and outputs in the process specification language, *NIST Internal Report 7152*, National Institute of Standards and Technology, Gaithersburg, MD.

Roberts, MJ and Newton, EJ: 2001, Understanding strategy selection, *International Journal of Human-Computer Studies* **54**: 137-154.

Schön, DA: 1983, *The Reflective Practitioner*, Harper Collins, New York.

Schön, DA and Wiggins, G: 1992, Kinds of seeing and their functions in designing, *Design Studies* **13**(2): 135-156.

Sim, SK and Duffy, AHB: 1998, A foundation for machine learning in design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **12**(2): 193-209.

Singh, P, Minsky, M and Eslick, I: 2004, Computing commonsense, *BT Technology Journal* **22**(4): 201-210.

Smith, GJ and Gero, JS: 2005, What does an artificial design agent mean by being 'situated'?, *Design Studies* **26**(5): 535-561.

Suwa, M, Gero, JS and Purcell, T: 1999, Unexpected discoveries and s-inventions of design requirements: A key to creative designs, *in* JS Gero and ML Maher (eds), *Computational Models of Creative Design IV,* Key Centre of Design Computing and Cognition, University of Sydney, Sydney, Australia, pp. 297-320.

von der Weth, R: 1999, Design instinct? – The development of individual strategies, *Design Studies* **20**(5): 453-463.

Ziemke, T: 1999, Rethinking grounding, *in* A Riegler, M Peschl and A von Stein (eds) *Understanding Representation in the Cognitive Sciences: Does Representation Need Reality?*, Plenum Press, New York, pp. 177-190.

FROM FORM TO FUNCTION: FROM SBF TO DSSBF

PATRICK W YANER AND ASHOK K GOEL
Georgia Institute of Technology, USA

Abstract. We describe a method of analogical reasoning for the task of constructing a Structure Behavior Function (SBF) model of a physical system from its drawing. A DSSBF (Drawing Shape Structure Behavior Function) model relates the SBF model of a system to its drawing. A DSSBF model of a target drawing is constructed by analogy to the DSSBF model of a very similar drawing. In this paper, we focus on the tasks of analogical mapping between target and source drawings and transfer of the DSSBF model of the source drawing to the target drawing. Archytas is a computer program that implements this analogical method for model construction in the domain of simple kinematics devices.

1. Motivation and Goals

Design is a prime example of situated cognition. Designers make decisions, solve problems, and construct models among other mental activities. However, their internal information processing is situated in the context of external artifacts ranging from physical to visual to mathematical. A central issue in design cognition thus is how do designers recognize, understand and make use of external representations? In this work, we examine a small piece of this very complex issue: how might designers construct mental models of physical systems from their drawings?

We consider 2D vector-graphics line drawings typically made by designers in the preliminary (conceptual, qualitative) phases of design as the input drawings to the model-construction task. As Figure 1 illustrates, an input target drawing specifies the form of a physical system in a 2D drawing. We assume a teleological model, and, in particular, a Structure Behavior Function (SBF) model, of the depicted system as the output of the task. The output specifies the function and the teleology of the physical system depicted in the input. We also assume that the designer is an expert in the design domain at hand, and thus has previously encountered numerous drawings and SBF models of them. Given this context, our hypothesis is that

J.S. Gero (ed.), Design Computing and Cognition '06, 423–441.
© 2006 *Springer. Printed in the Netherlands.*

designers might construct SBF models of physical systems from their drawings by *analogy* to SBF models of similar drawings.

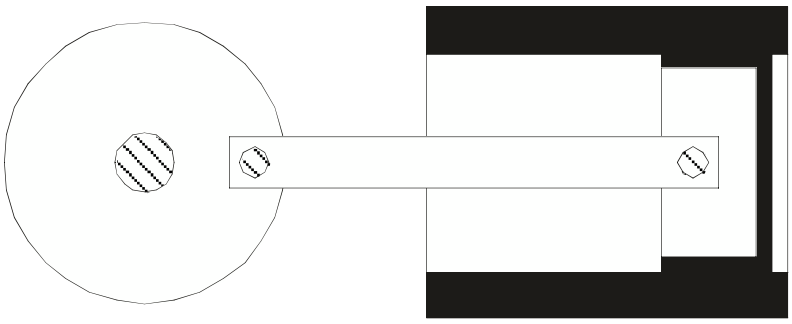


Figure 1. A sample input target drawing depicting the form of a physical system.

In general, analogical reasoning involves the steps of retrieval, mapping, transfer, evaluation, and storage. In this paper, we focus on the tasks of mapping and transfer. The design domain is that of kinematics systems, and, in particular, devices that convert linear motion into rotational motion (and vice versa). Archytas is a computer program that implements our theory of model construction by analogy, Figure 2. Since computer science already has developed a large number of techniques for extracting shapes and spatial relations from 2D vector-graphics line drawings, Archytas begins with a representation of shapes and spatial relations in an input drawing.

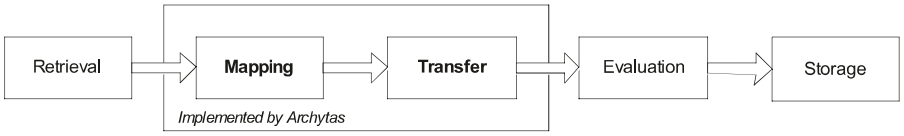


Figure 2. The steps of analogy. This paper—and our system, Archytas—deals with the mapping and transfer steps.

Both mapping and transfer are very complex tasks. The mapping task is complex because a given shape (or spatial relation) in the input target drawing may map into many similar shapes (or spatial relations) in the known source drawing. The transfer task is complex because based on a mapping between the shapes and spatial relations, the goal is to transfer the relevant portions of the structure, the behavior and the function (i.e., the SBF model) of the source drawing to the target drawing. Given this complexity, our work on the Archytas project so far has focused on analogical mapping and transfer between two *nearly identical* drawings. Specifically, in Archytas, we

assume that the target and the source drawings are so similar that any difference between them at the structural level of abstraction makes no difference at the behavioral and functional levels of abstraction. Figure 3 illustrates the source drawing for the target drawing of Figure 1.

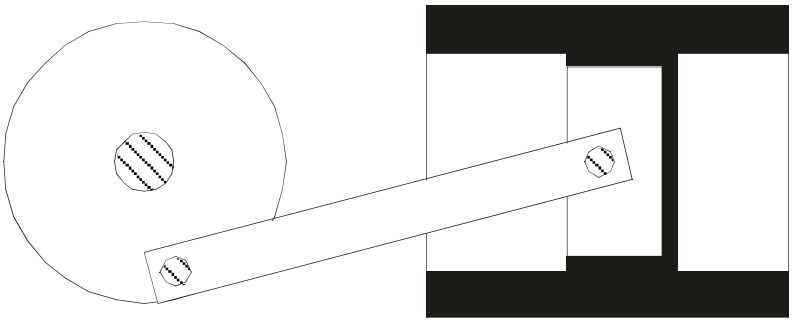


Figure 3. The source drawing corresponding to the target drawing of Figure 1.

Note that while the assumption of near identicality of the source and the target drawings simplifies the transfer task, it makes little difference to the mapping task. This raises the question of how can we control the complexity of mapping at the shape level? Our hypothesis is that knowledge of the functions of shapes in the source drawing informs the mapping task at the shape level, and seeds the mapping with shapes that play a critical role in the functioning of the device. The question then becomes how can functional knowledge about the shapes in the source drawing be represented, organized and accessed so that it can be used to guide mapping at the shape level? To address this question, we expand the schemata and extend the ontology of SBF models of physical systems into Drawing Shape Structure Behavior Function (DSSBF) models. Just as behavior is an intermediate abstraction between the structure and the function in an SBF model of a physical system, shapes and spatial relations are an intermediate abstraction between the structure and the drawing in a DSSBF model. The organization of DSSBF models enables our method for mapping to access the functions of the particular shapes in the drawing.

2. Background

This work intersects with three distinct lines of research: (1) SBF models of physical systems, (2) construction of models of physical systems from their drawings, and (3) analogical mapping.

2.1. SBF MODELS

The Kritik system (Goel and Chandrasekaran 1989, 1992; Goel 1991) was an autonomous case-based design system that addressed the task of preliminary (conceptual, qualitative) design of simple physical devices. It took a specification of the function desired of a device as input, and gave as output a specification of the structure of its design (i.e., a symbolic representation of the configuration of components and connections in the design). Since Kritik addressed the F ⇸ S (function to structure) design task, its design cases contained an inverse S ⇸ B ⇸ F (structure to behavior to function) mapping of the known designs, where the B in a S ⇸ B ⇸ F mapping stood for the internal causal behaviors that composed the functions of the components in the design into the functions of the design as a whole, Figure 4. Kritik's SBF model of a design represented function and behavior at multiple levels of aggregation and abstraction, and organized them into a F ⇸ B ⇸ F ⇸ B ⇸ F ⇸ ··· ⇸ B ⇸ F(S) hierarchy. Kritik showed that the SBF models provided a vocabulary for representing, organizing and indexing design cases, methods for retrieving and adapting known designs to meet new (but related and similar) functions, and methods for verifying and storing new designs in memory.

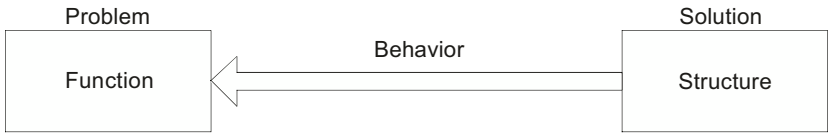


Figure 4. In an SBF model, the behavior of a physical system is an intermediate abstraction between its structure and its function.

The origin of Kritik's SBF models lies in Chandrasekaran's Functional Representation (FR) scheme for representing the functioning of devices (Sembugamoorthy and Chandrasekaran 1986; Chandrasekaran et al. 1993). In cognitive engineering, Rasmussen (1985) developed similar SBF models for aiding humans in trouble shooting complex physical systems. In computer-aided engineering, Tomiyama developed similar FBS models (Umeda et al. 1990) for aiding humans in designing mechanical systems. In design cognition, Gero et al. (1992) developed similar FBS models for understanding the mental information processing of designers in general. In their analysis of verbal protocols of designers working in a variety of domains, Gero and McNeil (1998) found that while (1) novice designers spend most of their time on the structure of the design solutions, spending relatively little time on the design functions or behaviors, (2) expert designers spend

significant amounts of time on all three major elements of FBS models: function, behavior, and structure.

2.2. MODEL CONSTRUCTION FROM DRAWINGS

Much of earlier work on constructing models from drawings typically has used forward-chaining rule-based reasoning. The early Beatrix system (Novak 1994) for example, used domain-specific rules to construct a structural model of simple kinematics systems (e.g., a block on an inclined plane) from a textually-annotated 2D diagram.

The more recent SketchIT system (Stahovich et al. 2001) takes as input a 2D sketch of a physical system, and gives as output multiple designs of the physical system in the kinematics domain, where each design is augmented with a simple state-transition diagram to describe the device's behavior. The system first produces a "behavior-ensuring parametric model" (or a BEP model) of the components of the design, and from this determines geometric constraints on the motion of the parts, generating all qualitative configuration spaces consistent with the behavioral constraints. Next, it selects motion types for each component, and, finally, from the motion types and the geometric interpretations provided by a library of interaction types, it generates a BEP model for the design as a whole.

GeoRep (Ferguson and Forbus 2000) takes as input simple a 2D vector-graphics line drawing depicting a physical process, e.g., a cup with steam coming out of it. It gives as output a symbolic description of the physical process depicted in the drawing, e.g., steam coming out of hot liquid contained in the cup. GeoRep is organized as a two-stage forward-chaining reasoner. First, a low-level, domain-independent relational describer recognizes shapes and spatial relations in the input line drawing, and then a high-level domain-specific relational describer applies domain-specific rules to produce an final description of the physical process in the diagram. In contrast to all of these, our work derives structure from shape by analogy.

2.3. ANALOGICAL MAPPING

Analogical mapping between a source case and a target problem can be viewed as a graph isomorphism problem, in which both the source and the target are represented as labeled graphs, and the goal is to find correspondences (similarities and differences) between the elements (vertices and edges) of the two graphs. An individual *map* associates individual elements (vertices or edges) in the graphs representing the source and the target, whereas a *mapping* associates two subgraphs and is composed of many individual *maps*. The general graph isomorphism problem, when dealing with subgraphs, is computationally intractable (NP-Hard).

The Structure-Mapping-Engine (SME) (Falkenhainer et al. 1990) is a powerful, but content-free, analogical mapping system. SME first generates local maps between the target and the source graphs, then uses heuristics based on the structure of the graphs to select among the local maps, and finally to builds a consistent mapping. JUXTA (Ferguson and Forbus 1998) uses SME to compare two nearly identical drawings of a physical process, such as two drawings of a coffee cup with a metal spoon, but with a thicker metal bar in one drawing than in the other. JUXTA first uses GeoRep for deriving structure from shape, and then uses SME to compare the two structures, looking for alignable differences, and drawing candidate mappings between the two drawings based on these differences. In contrast, our work uses functional knowledge to guide the process of analogical mapping between two nearly identical drawings at the shape level.

The Analogical Constraint Mapping Engine (ACME) (Holyoak and Thagard 1989) views both analog retrieval and mapping as constraint satisfaction tasks, where the constraints to be satisfied can be structural, semantic, and pragmatic. ACME uses a relaxation scheme implemented in a recurrent localist neural network with each individual map between the target and the source corresponding to a network node. ACME returns the mapping whose individual maps have the highest activation at the end. Geminus (Yaner and Goel 2004) uses a symbolic constraint-satisfaction method for analogical retrieval and mapping. It deals only with structural constraints, and uses subgraph isomorphism as the similarity metric for retrieval.

In computer-aided design, Gross and Do's Electronic Cocktail Napkin (Do and Gross 1995) uses a simple count of matching elements between the source and target drawings for analogical retrieval. TOPO (Börner et al. 1996), a subsystem of FABEL (Gephardt et al. 1997), found the maximum common subgraph between the source and the target drawings for retrieval. In this work, we use the closely related criterion of *maximum overlap set* (Chen and Yun 1998), which is also known as *maximum common edge subgraph* (Raymond et al. 2002), for analogical mapping. The main point of our work, however, is the use of functional knowledge of shapes in the source drawing to seed the mapping process. In principle, one could use this functional knowledge in conjunction with a different mapping technique, such as that of SME.

3. DSSBF: A Unified Form-Function Model

A DSSBF model of a physical system unifies the functional and spatial representations of the system. In a DSSBF model, shapes and spatial relations are an intermediate abstraction between the structure and the drawing in a DSSBF model. Just as SBF regards behavior as mapping structure onto func-

tion, Figure 4, DSSBF regards shape as mapping drawing (form) onto structure, Figure 5.

Note that structure occurs in both Figure 4 and Figure 5, and forms the links between the spatial and the functional representations of a system. This results in a five-level model with function at the top and form (e.g., a drawing) at the bottom, with shape, structure and behavior as intermediate levels of abstraction, Figure 6. Note that following the F ⇉ B ⇉ F ⇉ B ⇉ F ⇉ ··· ⇉ B ⇉ F(S) organization of SBF models, in general the DSSBF model may have multiple functional and behavioral levels (but Figure 6 does not depict this for clarity). Note also that in general there may be many drawings, and, hence, many shape representations, of a single SBF model of a device design (but again Figure 6 does not depict this for clarity).

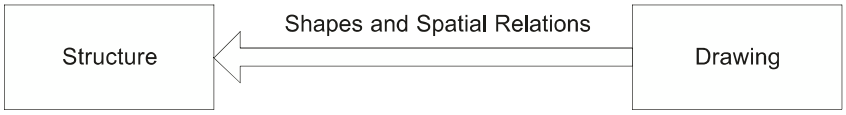


Figure 5. In DSSBF model, shape is an intermediate level of abstraction between the drawing of a physical system and its structure.

The representations of any two consecutive levels in the five-level DSSBF model contain two-way pointers to each other. For example, as in SBF models, the specification of a function specifies the behavior that accomplishes it and the specification of a behavior specifies the function (if any) that it accomplishes. Similarly, the specification of a behavior specifies the structural constraints (in the form of connections among components) that enable it, and the specification of a component specifies its functional abstractions and the role they play in a behavior. In addition, in a DSSBF model, the specification of a structural component or connection specifies the shape that depicts it in a drawing, and the specification of a shape specifies the component or connection it depicts. Thus, the organization of a DSSBF model of a physical system affords navigation of the entire model, and accessing of knowledge at one level of abstraction that is relevant to reasoning at another level.

4. DSSBF: An Illustrative Example

The kinematics system shown in Figures 1 and 3 is a piston and crankshaft. In this device, there are five components (though only the four are depicted in these two Figures): piston, crankshaft, connecting rod, cylinder, and crankcase (not depicted).

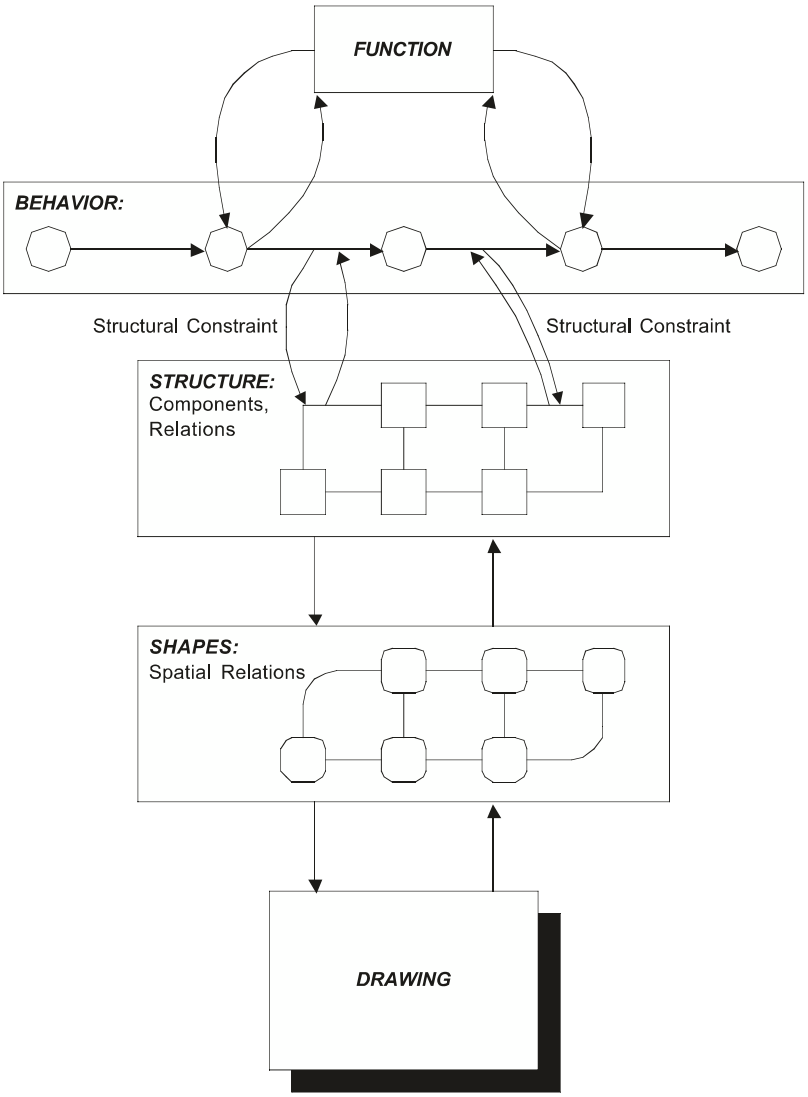


Figure 6. The schemata of Drawing Shape Structure Behavior Function (DSSBF) models. This diagram shows the organization of DSSBF models, with function at the top and form (a drawing) at the bottom.

***Function*:** The function of this system is to turn the crankshaft. Figure 7 illustrates a (partial) specification of this function in the DSSBF language. The function is specified as a schema, in which the "Given" and "Makes" slots refer to a pair of states: the given condition which must be true for the system to work, and the condition which the system should achieve. The "By" slot contains a pointer to the behavior that accomplishes this function.

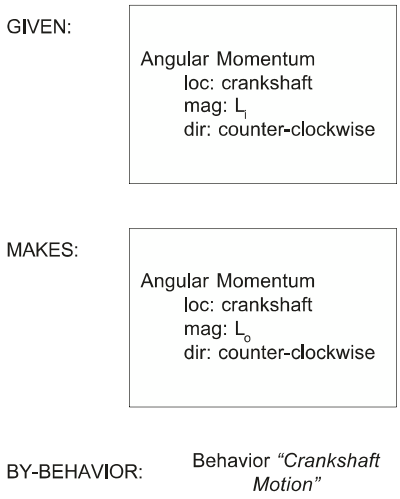


Figure 7. Partial specification of function in the DSSBF model.

***Behavior*:** In a DSSBF model, a behavior (i.e., an internal process) is specified as a sequence of discrete states and state-transitions between the. The states in a behavior specify values of system variables relevant to the behavior. The annotations on a state transition specify causes and the conditions of the transition. Figure 8 illustrates a (partial) specification of the behavior of the crankshaft. The behavior tracks the angular momentum of the crankshaft, which it gains from a downward force coming from the connecting rod through the joint with the connecting rod, and loses through friction. Note that illustration of the crankshaft behavior shows three states in linear succession, but the third state is a repetition of the first, so that the behavior loops. Note also that one of annotations on the first state transition in the crankshaft behavior refers to the behavior of the connecting rod. The behavior of the connecting rod is represented similarly.

***Structure*:** In a DSSBF model, structure specifies the components and the connections among the components. Table 1 shows an outline of the specification of components. Briefly, each component has properties, which take values, and quantities, which have a type of either scalar or vector, and which are variables whose values are changed by the causal processes in the behaviors of the system.

Connections are represented as schemas. Connections also have types indicating their degrees of freedom, if any (revolute, prismatic, fused or adjoined, and so on). Figure 9 illustrates the connections in the piston and crankshaft example.

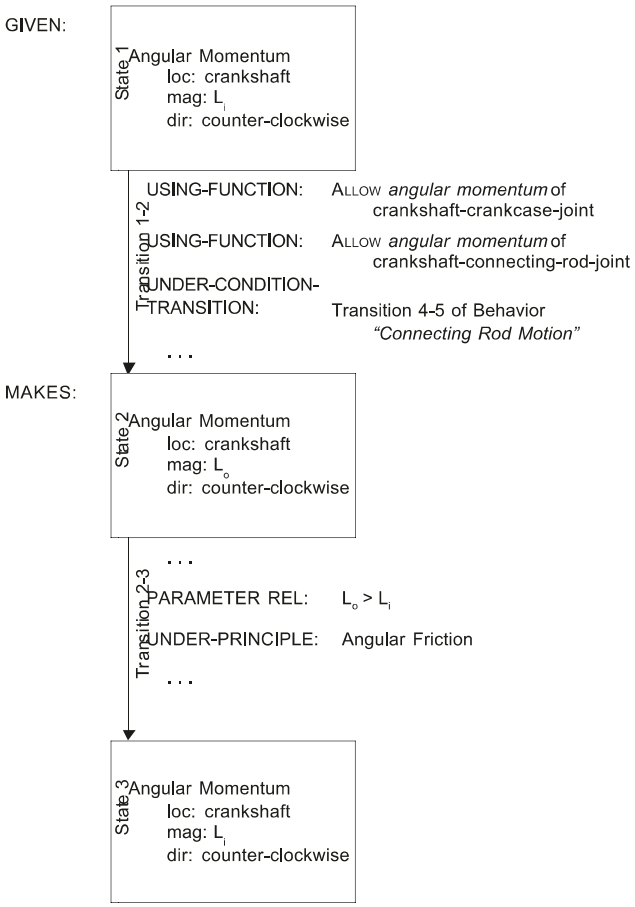


Figure 8. Partial specification of the crankshaft behavior in the DSSBF model.

TABLE 1. Partial Specification of Components in the DSSBF model.

Component	Properties	Variable Quantities	Connected to
Piston	height, diameter	linear momentum	cylinder, connecting rod
Crankshaft	diameter, mass	anglular momentum	crankcase, conn. rod
Connecting Rod	length	ang. & linear momentum	crankshaft, piston
Cylinder	diameter, length		piston, crankcase
Crank case			cylinder, crankcase

4.1. SHAPES AND SPATIAL RELATIONS

Let us consider the shape-level representation of the drawing of the source case illustrated in Figure 3. Since this is a vector-graphics file, the properties and locations of lines and their interconnections already is known. Even the

fact that one rectangle is overlapping another (which might otherwise be slightly tricky to detect from nothing but a 2D depictive representation) can be assumed as given. Thus, we have whole shapes, such as rectangles and circles, and their geometric properties, but we need to know what the relevant interrelationships among the shapes are.

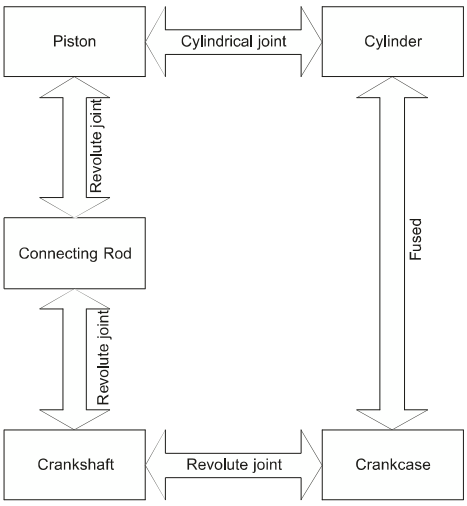


Figure 9. The connections between the components described in Table 1.

For vector-graphics drawings such as Figure 3, DSSBF models uses a taxonomy of spatial relationships among the shapes in a drawing:

- Parallel-ness perpendicularity
- End-to-end and overlapping connections between lines
- Collinearity
- Horizontal and vertical alignment and relative length
- Containment

In reference to Figure 3, first, parallel-ness and perpendicularity are important; the two rectangles representing the cylinder are parallel to each other. Connectivity, too, is important, as the rectangle representing the piston shares sides with the cylinder's rectangles, and the connecting rod overlaps with the piston and also the cylinder's circle. Next, alignment is important, as the center of the circle lines up with the center of the piston and cylinder, and the two cylinder rectangles are vertically aligned. Relative length too is important, as the piston is shorter than the cylinder (as it must move within it). Finally, containment, is important as the circles representing joints are contained within larger shapes representing the connected components (there

are three in the drawing: (1) the piston/connecting rod joint, (2) the crankshaft/connecting rod joint, and (3) the crankshaft/crank case joint).

Figure 10, illustrates the representation of spatial relations using the above spatial relations among the shapes in the drawing of the source case, Figure 3. This representation is somewhat abbreviated, as, for instance, the component lines of the rectangles and the part-whole relations with the rectangles, as well as the interrelationships between them, are not shown.

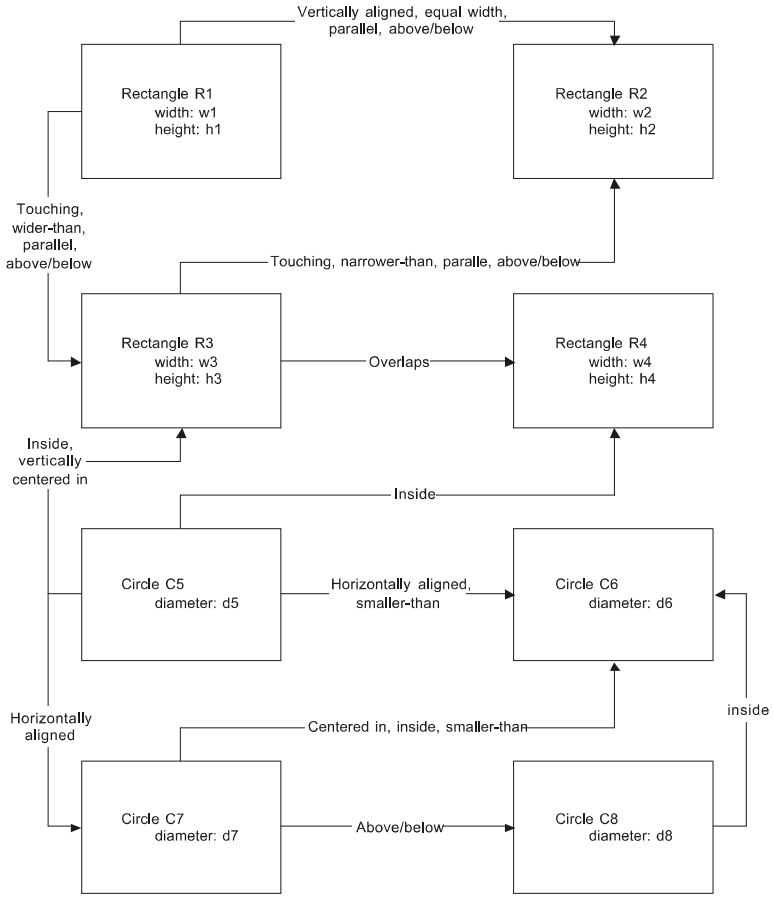


Figure 10. Representation of some the shapes and some of the spatial relations for the source case, Figure 3. This diagram does not show the aggregation of lines into rectangles, and their interrelationships—only the properties and relations of the aggregated shapes.

In contrast, the representation for the target drawing illustrated in Figure 4 would have some relationships different, but most of the relationships would be the same because the two drawings are nearly identical. The big-

gest difference would be the parallel-ness of the connecting rod with the cylinder, and the relative orientation of the crankshaft bearing with the crankshaft/connecting-rod joint (vertical in the source, horizontal in the target).

4.2. RELATING DRAWING AND SHAPES TO STRUCTURE

In order to be useful for analogical mapping and transfer, the representation of the shapes and spatial relations of the drawing need to be related with the structural elements in the DSSBF model. In general, these relations take the form of links between the shapes shown in Figure 11 and the component and connection schemas outlined in Table 1 and Figure 9, respectively.

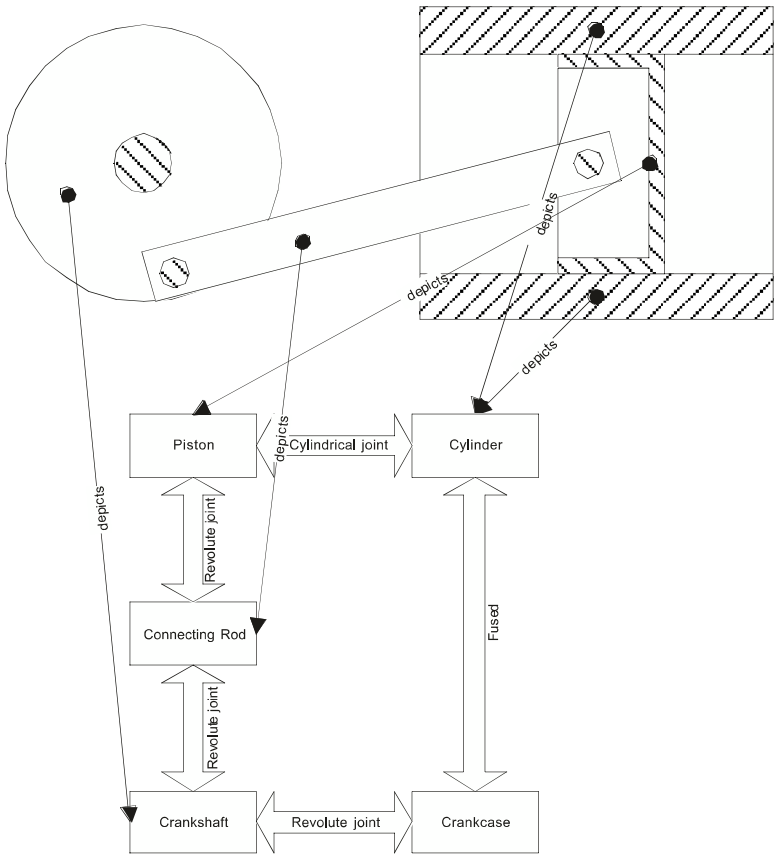


Figure 11. Linking shape to structure in a DSSBF model. The source drawing in Figure 3 is associated shape-by-shape with the structural elements as shown in Table 1 and Figure 8. Although this Figure shows the drawing itself, in fact each shape element from the shape representation, Figure 10, is linked to a component or connection.

At the most basic level, there is a relation of the form "*A* depicts *B*" (*A* being a shape schema, *B* being a component schema) from the shapes to the components. It is important to note that only the shapes themselves enter into these "depicts" relations; the spatial relations between shapes do not.

5. Mapping: Recognizing Form by Using Function

We want to build an account of the process of recognizing a target drawing of an as-yet unknown physical system by analogy to a nearly identical source drawing of a known physical system for which we have a complete DSSBF model. We have developed an algorithm for generating all partial mappings between the shape-level representations of a source and a target drawing. Both the source and the target drawings are represented as labeled graphs, with shapes as the vertices and the spatial relations as edges among them. The algorithm collects individual *maps* from shapes in the source to shapes in the target, and attempt to merge them into whole *mappings* from the source to the target. The algorithm computes partial mappings, where some of the relations do not necessarily match. This corresponds to the problem of *maximal common edge subgraph*, also known as *maximal overlap set.* A maxim*al* mapping is one that cannot be expanded to include other terms without contradiction, and a maxim*um* mapping is the largest of all of the maxim*al* mappings. We use an algorithm that lists all maxim*al* mappings, choosing the largest of these, the maxim*um* mapping, as the result.

We begin by marking certain of the shapes in the source as "important"—a form of pragmatic marking—so that no mapping is returned that does not involve at least one of these relations. The important shapes are those that play a critical role in functioning of the physical system depicted in the drawing. In the piston and crankshaft example, since the function of the system is to turn the crankshaft, the shape of the crankshaft is marked as important, as are the shapes depicting the piston and connecting rod. The algorithm can determine this importance by navigating the DSSBF model of the system in the source drawing. Also, a minimum bound is chosen for the size of the subgraphs/mappings, so that degenerate cases of only a single term in a large image are not returned (there can be dozens or even hundreds of these, even when there's a single complete mapping).

The procedure, at an abstract level, runs as follows:

1. Gather all maps between source and target relations. Each map between a pair of relations will entail *two* maps between the entities related, so that, if "*A* is-left-of *B*" maps to "*X* is-left-of *Y*", then this entails *A* maps to *X* and *B* maps to *Y*.
2. Those maps involving marked ("important") relations are set aside as "important" maps.

3. Choose one of the "important" term maps M_1. Now, gather every other map M_i for $i>1$, such that M_1 and M_i are consistent. They are consistent when:
   - the source terms in M_1 and M_i are different
   - the target terms in M_1 and M_i are different
   - the associated entity maps are consistent (same source entity maps to the same target entity, and conversely the same target entity has the same source entity being mapped to it; or else different sources, different targets)

   These rules enforce a one-to-one mapping between both relations and entities.
4. When all of the mutually consistent mappings have been gathered, save this as a single (partial) mapping if it exceeds the minimum size for an acceptable mapping (minimum bound).
5. Choose the next marked "important" term map M_2, and repeat steps 3 and 4, iterating through all of the marked "important" term maps.
6. Return the resulting list of mappings.

This mapping algorithm allows term-by-term comparison of the source and target, so that the similarities and differences between them with respect to a potential alignment of the drawings can be employed and reasoned about. The algorithm returns all maximal complete mappings, and, as we've said above, the largest of these is the maximum mapping.

In the context of the target and source drawings of Figures 1 and 3, respectively, we would expect all the shapes to map onto each other, but the algorithm discovers that the target, Figure 1, has the connecting rod rectangle parallel with the cylinder and the piston, but the source, Figure 3, does not. Note that it will also return several partial mappings: for instance, the top rectangle for the cylinder in Figure 1 may map to the bottom rectangle for the cylinder in Figure 3 or the top one. These are inconsistent with each other, but both would be maximal, and both would be returned.

6. Transfer: Interpreting Form by Constructing Functional Model

Once we have recognized a drawing by mapping its shapes and spatial relations onto that of another nearly identical drawing, the next task is interpretation of the drawing. This is accomplished by transferring the DSSBF model of the source drawing to the target drawing, proceeding forward from shape through structure and behavior to function. An outline of the procedure is as follows:

1. Some of the shapes in the source drawing may be grouped together, if they, as a group, depict a single component (such as the pair of rec-

tangles in Figure 3 depicting the cylinder). For these shapes, when they are mapped to target shapes, transfer these groupings, as well, from source to target.

2. Each shape in the source drawing either is related to a component or connection via a "depicts" relation (e.g. *A* depicts *B*, where *A* is a particular shape from the drawing and *B* is a component from the structural model), and components and connections that are not depicted are marked as "undepicted" in that drawing. Transfer each of these "depicts" and "undepicted" relations to the target drawing.
3. These "depicts" relations set up the elements of the structural model in the target, which can now be mapped from the source, and thus each property and quantity of the named components is transferred from source to target.
4. The connections between these newly transferred components can then, themselves, be transferred from source to target, extending the mapping.
5. Certain components are involved in behaviors, and are thus linked by a relation ("*B* of individual *C*", where *B* is a behavior and *C* is a component). These relations are then transferred, setting up the behavioral model to be transferred as well.
6. The behavioral model is transferred by iterating through the named behaviors in the target and transferring the states, transitions, and all properties and relations thereof from source to target.
7. Finally, some states and behaviors are named in the functional specification of the device. Following the links from the behavior to the function in the source, transfer the functional specification from source to target.

Figure 12 illustrates the use of this procedure for transfer of the structural model in the piston and crankshaft example. Since in this example the shape level differences between the source and target drawings make no behavioral difference, transferring the behavior and functions is trivial.

7. Conclusions

In this paper, we examined two related issues: (1) how might a designer recognize a target drawing by comparing it to a similar source drawing, and (2) how might a designer interpret a target drawing by constructing a functional model of it? We described a method for analogical mapping between the target and a nearly identical source drawing, and a method for transferring the functional model of the source drawing to the target drawing when the differences between the drawings make no functional difference. The functional model of the source drawing is an integrated form-function model called the DSSBF model. The organization of the DSSBF model allows ac-

cess to the specification of function of shapes in the source drawing, and to use the shapes that have important functions to seed the mapping algorithm.

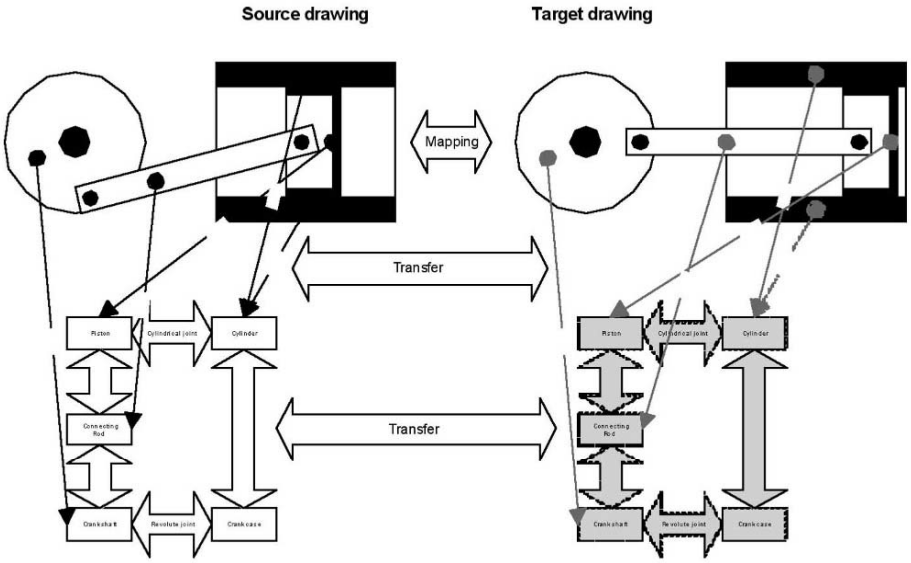


Figure 12. The process of analogical mapping and transfer illustrated. The source drawing (in fact its shape representation) at left is mapped onto the target, at right. The structural model (from Figure 8) and the links from shapes to structural elements are all transferred from source to target.

This work however has several limitations. The most obvious limitation is that so far we have evaluated our method for mapping only for situations in which the target drawing is nearly identical to the source drawing, and our method for transfer is further limited to situations in which the differences in drawings make no difference no functional difference. Another obvious limitation is that if the target drawing were made from a different view (e.g., the top view), then our method would break down even if the target drawing in fact was representing exactly the same physical system in exactly the same state as the source drawing. Yet another limitation is that the drawings in our work are 2D vector-graphics line drawings. Thus, in its current form, our work represents only a small first step towards building a computational theory of model construction from drawings by analogy.

Acknowledgements

This research has been supported in part by a NSF (IIS) grant (Award number 0534266) on Multimodal Case-Based Reasoning in Modeling and Design. This paper has significantly benefited from critiques by anonymous reviewers of an earlier draft.

References

Börner, K, Eberhard, P, Tammer, E-C, and Coulon, C-H: 1996, Structural similarity in adaptation, *in* I Smith and B Faltings (eds), *Lecture Notes in Artificial Intelligence*, Springer-Verlag, **1168**: 58-75.

Chandrasekaran, B, Goel, AK, and Iwasaki, Y: 1993, Functional representation as design rationale, *IEEE Computer* **26**: 48-56.

Chen, CK and Yun, DYY: 1998, Unifying graph matching problems with a common solution, *Proc. Int'l Conf. on Systems, Signals, Control, and Computers*.

Do, EY-L, and Gross, MD: 1995, Drawing analogies: finding visual references by sketching, *Proc. Association of Computer-Aided Design in Architecture, National Conf.*, ACADIA, pp. 35-52.

Falkenhainer, B, Forbus, K and Getner, D: 1990, The structure mapping engine: Algorithm and examples, *Artificial Intelligence* **41**: 1-63.

Ferguson, RW, and Forbus, KD: 1998, Telling juxtapositions: using repetition and alignable difference in diagram understanding, *in* K Holyoak, D Gentner, and B Kokinov (eds), *Advances in Analogy Research*, New Bulgaria University, Sofia, Bulgaria, pp. 109-117.

Ferguson, RW, and Forbus, KD: 2000, GeoRep: A flexible tool for spatial representation of line drawings, *in Proc. 17th National Conf. on Artificial Intelligence (AAAI-2000)*, AAAI Press, pp. 510-516.

Gephardt, F, Voss, A, Grather, W, and Schmidt-Belz, B: 1997, *Reasoning with Complex Cases*, Kluwer Academic Publishers, Dordrecht.

Gero, J and McNeil, T: 1998, An approach to the analysis of design protocols, *Design Studies* **19**: 21-61.

Gero, J, Tham, K, and Lee, S: 1992, Behavior: a link between function and structure, *in* DC Brown, MB Waldron, and H Yoshikawa, (eds), *Intelligent Computer-Aided Design*, North-Holland, pp. 193-225.

Goel, AK: 1991, A model-based approach to case adaptation, *Proc. 13 Annual Conf. of the Cognitive Science Society*, Lawrence Erlbaum Associates, pp. 143-148.

Goel, AK and Chandrasekaran, B: 1989, Functional representation of designs and redesign problem solving, *Proc. 11th Int'l Joint Conf. on Artificial Intelligence (IJCAI-89)*, Morgan Kaufmann, pp. 1388-1394.

Goel, AK and Chandrasekaran, B: 1992, Case-based design: a task analysis, *in* C Tong and D Sriram, (eds), *Artificial Intelligence Approaches to Engineering Design, Volume II: Innovative Design*, Academic Press, San Diego, CA, pp. 165-184.

Holyoak, K and Thagard, P: 1989, Analogical mapping by constraint satisfaction, *Cognitive Science* **13**: 295-355.

Rasmussen, J: 1985, The role of hierarchical knowledge representation in decision making and system management, *IEEE Transactions on Systems, Man, and Cybernetics* **15**: 234-243.

Raymond, JW, Gardiner, EJ, and Willett, P: 2002, Heuristics for similarity searching of chemical graphs using a maximum common edge subgraph algorithm, *J. Chem. Inf. Comput. Sci.* **42**: 305-316.

Sembugamoorthy, V and Chandrasekaran, B: 1986, Functional representation of devices and compilation of diagnostic problem-solving systems, *in* J Kolodner and C Riesbeck (eds), *Experience, Memory, and Reasoning*, Lawrence Erlbaum, Hillsdale, NJ, pp. 47-73.

Stahovich, TF, Davis, R, and Shrobe, H: 2001, Generating multiple new designs from a sketch, *Artificial Intelligence* **104**: 211-264.

Umeda, Y, Takeda, H, Tomiyama, T, and Yoshikawa, H: 1990, Function, behavior, and structure, *Proc. 5th Int'l Conf. on Applications of AI in Engineering*, vol. 1, Springer-Verlag, pp. 177-193.

Yaner, P and Goel, A: 2004, Visual analogy: reexamining analogy as a constraint satisfaction problem, *Proc. 26th Annual Meeting of the Cognitive Science Society*, pp. 1482-1487.

FORMAL DESCRIPTION OF CONCEPT-SYNTHESIZING PROCESS FOR CREATIVE DESIGN

Taxonomical relation and thematic relation

YUKARI NAGAI
Japan Advanced Institute of Science and Technology, Japan

and

TOSHIHARU TAURA
Kobe University, Japan

Abstract. We describe a design synthesizing process which has been pointed out to be a key to creative design. We describe two topics of design study. First, from the perspectives of creativity, the concept-synthesizing process is formed with the 1st primitive of the concept-synthesizing process being 'concept abstraction' with the principle of 'similarity' in 'taxonomical relations', the 2nd primitive being 'concept blending,' in which the principle is 'similarity' and 'dissimilarity' in 'taxonomical relations', and the 3rd primitive being 'concept integration' and with the principle of 'thematic relations'. Second, design experiments using protocol analysis were conducted to identify what/how design primitives are related to higher creativity. As a result, in the process of synthesizing concepts, thematic relations between two concepts significantly extend the design space, which led to higher creativity. Given this, the creative design process can be driven by the 3rd primitive of the concept-synthesizing process.

1. Introduction

Many studies have been conducted to analyze the characteristics of the design thought process from the viewpoint of creativity. As a result, it has been found that concept-synthesizing processes, such as combining, blending or integrating two different concepts, are keys to creative thinking. Analogical reasoning and metaphor are known to play very important roles in creative design (Gero and Maher 1991; Goldschmidt 2001). For example, the 'Swan chair' is a famous example of design, which had been imaged using analogy. Its form resembles a swan, and users understand its message

J.S. Gero (ed.), Design Computing and Cognition '06, 443–460.
© 2006 *Springer. Printed in the Netherlands.*

of 'this chair is soft and elegant like a swan'. Figure 1 shows some examples of design analogy. Chairs designed using analogical reasoning resemble a swan, a mushroom and a helicopter (Swan chair 1958; Mushroom stool 2003; Easy chair 2000). Figure 2 shows a sample of a product designed using metaphor. Its message is that 'this is a new object that will induce mellow feelings in your daily life'.

From the viewpoint of mental cognition in the domain of cognitive science, Finke et al. (1992) described conceptual synthesis as an efficient means of developing creative insights into new inventions, and carried out experiments on creation as mental products generated by imagery synthesis. For supporting human creativity, it has been pointed out that it is significant to develop creative thinking that is related to the transforming of concepts (Boden 2004). Creative thinking is comprehended as conceptual space (Gardenfors 2000). On the other hand, in studies on cognitive linguistics, Fauconnier (1994) focused on the construction process of meaning in ordinary discourse. He analyzed how conceptual integration creates mental products and how to deploy systems of mapping and blending between mental spaces. From the viewpoint of mental space theory, he showed that conceptual integration operates on two input mental spaces to yield a third space which is called 'the blend'. That blended space inherits partial structures from the input spaces and has emergent structures of its own. Both mental products, imagery and discourse, have shown emergent features and they have stimulated creativity. Fauconnier and Turner (2002) suggested that a watch is designed by conceptual blending.

Although it has been pointed out in many studies that synthesizing two concepts is the key to creative design, these concept synthesizing processes have not yet been formerly described, and the kinds of primitives and how these primitives are related to creativity have not been clarified. In order to gain a deeper understanding of the nature of creative design and to develop methodologies for creative design, it is important to determine primitive processes for concept-synthesis. We assume that primitive processes are useful for explaining creativity in design, rather than a general process model in which only the superficial design action is generalized and the hidden thought mechanism is not dealt with.

Normally, an 'abstraction process' based on a 'taxonomical relation' is regarded as a primitive process in creating a new concept. In addition, another important process for recognizing two concepts is pointed out. It is called the integrating process, in which two concepts are related thematically. For example, from the two concepts, milk and cow, a scene of milking a cow can arise from the thematic relating process. This process is expected to be effective for creative design. However, how the thematic relation is effective for design creativity has not been clarified.

In this paper, we describe two topics. First, the concept-synthesizing process (combining, blending, and integrating) is formed from the viewpoint of creativity. Second, the relationships between creativity and design primitive processes, focusing particularly on the relation types- taxonomical relation or thematic relation - are empirically studied.

Figure 1. Swan chair (left), Mushroom stool (center) and Easy chair (right).

Figure 2. 'Sound Object' designed by Anna Von Schewen (2002).

2. Form Description of Synthesizing Process

2.1. CONCEPT ABSTRACTION

Analogical reasoning and metaphor are understood to be methods of concept creation via the transfer of a new concept from an existing concept. In practice, they are frequently used in the design process. For example, `designing a musical instrument like a dress' is one way of creating a new concept of a musical instrument. We can imagine many new instruments in this way by using metaphors, for example, 'an instrument like a harp', Figure 3. In this thought process, the design result (a musical instrument) is designed such that it and a dress share some common features, such as shape

and function. Generally speaking, the primitive process of recognizing common features is the 'abstraction process' based on 'taxonomical relation (explained in Section 2.3)' focusing on the 'similarity' between two things. Therefore, the 1st primitive of the concept-synthesizing process is 'concept abstraction,' and its principle is 'similarity' in 'taxonomical relations.'

Figure 3. An idea designed by using metaphor.

2.2. CONCEPT BLENDING

Although we recognize that analogical reasoning and metaphor are powerful for generating a new concept, we suspect that there is a more creative design method because the main roles of analogical reasoning and metaphor are to understand or to transfer a known concept; that is, it is analytic rather than synthetic since its primitive process is the extraction of some features from a known concept by analyzing it.

We can think of a concept-blending process as that in which two basic concepts are blended at an abstract level and a new concept that inherits some abstract features of the two base concepts but concrete features of neither are generated. For example, 'design something by combining the concepts of a musical instrument and a dress,' where the design result could be a guitar, the outside and sound of which can be changed to suit the surroundings like changing a dress, or a melody costume, that is, a wearable musical instrument. Another example is a wine glass which induces melody by blending a concept of party and that of strings, Figure 4. This concept-blending process seems to be similar to analogical reasoning or the metaphor process. However, these two processes are different in the following points. In the case of analogical reasoning, the harp, a musical instrument, is predicted to induce dressy feelings of elegance and distinction. Therefore, the harp is a medium and the dress is an intention similar to a relationship

between sign and meaning in semiotic relations. Also, in the metaphor process, a musical instrument again has the role of a medium to give the meaning of dress. In both cases, the roles are the same. In contrast, the relationship between a musical instrument and a dress in the concept-blending process is different. One does not express the other. The new concept is not just the medium of an instrument nor is it a dress. It has no strong association with the two base concepts. Therefore, it presents a high possibility of creating a novel concept. In the concept-blending process, not only 'similarity' but also 'dissimilarity' is pointed out, since the specific features belonging to each individual concept are blended. Therefore, the 2nd primitive of the concept-synthesizing process is 'concept blending' and its principle is 'similarity' and 'dissimilarity' in 'taxonomical relations.'

Figure 4. An idea designed by concept blending.

2.3. CONCEPT INTEGRATION

In the research on recognizing the relation between two concepts, it is pointed out that there are two kinds of relations (taxonomical relation and thematic relation) between two concepts. Wisniewski and Bassok (1999) studied the relative tendency to use comparison versus integration in making similarity judgments by orthogonally varying pairs of objects so as to be taxonomically or functionally related. As a result, it was shown that not only a taxonomical relation but also a thematic relation is important in recognizing the two objects. The former is a relation that represents the physical resemblance between the two objects, for example, "milk and coffee are drinks." The later is a relation that represents the relation between two concepts through a thematic scene. For example, a scene of milking a cow is recollected from the two concepts of milk and cow. In such a sense, milk and cow are related to each other. In this kind of thematic relation, a

dress is not physically related to a musical instrument but people imagine a scene in which a dressy lady plays the violin.

In design, the result (product) must be meaningful to people. Therefore, the designer must carefully consider not only its attributes (shape, material, etc.) but also its function and interface with the user, that is, consideration of the human factor is important. Recognizing objects in a thematic relation is to recognize them from the human viewpoint. Consequently, the thematic relation is expected to be closely related to design creativity.

Therefore, the 3rd primitive of the concept-synthesizing process is 'concept integration' and its principle is 'thematic relations.'

We summarize the formal description of the concept-synthesizing process in design in Table 1.

TABLE 1. Three kinds of design process primitives and principles.

	Design Process Primitive	Principle
1st	Concept Abstraction	taxonomical relation (similarity)
2nd	Concept Blending	taxonomical relation (similarity and dissimilarity)
3rd	Concept Integration	thematic relation

3. How Design Principle Affects the Design Creativity

How the design principle (taxonomical relation or thematic relation) affects the design creativity is clarified using both the design results and the thought process, focusing on the extension of idea space.

Also, the 2nd and 3rd design process primitives are made to relate more closely to a higher creative process in design than the 1st. In this research, we focus on the 2nd and 3rd primitives, with emphasis on the concept-synthesizing process caused by different types of relations – taxonomical or thematic - between two concepts.

3.1. METHODS

To elucidate the structuring process of design ideas, analyzing not only the design outcomes but also the design processes, that is, design thinking process and midterm representations, provide crucial keys (Lawson 1997). In this research, a design experiment is performed, and not only the design results but also the process of design thinking are analyzed. In particular, the

difference in the extensions of design spaces in the concept-synthesizing process, focusing on the extension process of idea space of the subjects and the effect of the difference in the relationships (in taxonomical relations for the 2nd primitives, or in thematic relations for the 3rd primitive) on creativity is analyzed in this study.

3.2. ANALYSIS OF DESIGN PROCESS

In this research, protocol analysis and semi-structured interviews are implemented. The think aloud method is adopted for acquiring utterances as protocol data for designing (Ericsson and Simon 1984). In this method, the subjects are requested to say aloud what they are thinking while performing a task. The utterances are recorded and the data are analyzed. In order to identify which relationship between two concepts the subject considered, the reason behind the design idea is examined. However, it is difficult to obtain data on such reasons, because the subjects do not always state the reasons behind their thinking. Therefore, in this research, the method of protocol analysis based on the explanation of design activities is adopted (Taura et al. 2002).

3.3. CREATIVITY EVALUATION OF DESIGN RESULT

The design results are evaluated based on the method of Finke et al. (1992), that is, from the two viewpoints of practicality and originality, on a five-point scale.

3.4. METHOD OF EXPERIMENT

In this research, with the aim of examining the conceptual synthesizing process, the design experiment is conducted focusing on the extension process of idea space which is formed through design space blending (Taura et al. 2005).

We analyze the design thinking process from the following two perspectives.

- From the macroscopic perspective, does the design process involve thematic integration or taxonomical blending?
- From the microscopic perspective, is the design process associated with thematic relations or taxonomical relations?

The experiment is composed of two parts, the design session and the interview session.

3.4.1. Design task

The subjects were asked to perform two kinds of design tasks at random. Base concepts were selected based on the research of Wisniewski and Bassok (1999).

- Task 1: Design new furniture starting from the word "Cat-hamster"
- Task 2: Design new furniture starting from the word "Cat-fish"

The reason for showing the synthesized word as "Cat-hamster" and "Cat-fish" is that the subject will be able to understand the idea of "conceptual blending" easily (Harakawa et al. 2005).

3.4.2. Method of experiment

The design experiment is structured as follows.

1) Design session (10 minutes)
The subject is made to perform the design task by the think-aloud method, and the utterances and the sketch are recorded with a tape recorder and a video camera. The purpose of this session is to obtain the protocol data and the sketch.
2) Interview session (30 minutes)
The subject is asked to explain the reason for each design activity while monitoring the video of the design session. The purpose of this session is to determine the reasons why new concepts were generated (Questionnaires; 'where did it come from?', 'why did you draw this shape?', and so on).
3) Creativity evaluation
The design results are evaluated based on the two viewpoints of practicality and originality on a five-point scale. Only the designs with more than 3 practicality points are evaluated from the viewpoint of originality.

3.5. RESULT OF DESIGN EXPERIMENT

The design experiment was conducted with three subjects. In total, fifteen design ideas were presented. Because the subjects were not experienced designers, creativity was evaluated on the basis of the design concept. The experimenter prepared design concept summaries on the basis of the design idea and the interview of the subject. Fifteen design concepts for two tasks (No.1-15) are shown below as the design results.

• Task A: Design new furniture starting from the term "Cat-hamster"

Design result 1

'A wardrobe with pet rooms'

There are rooms for the cat and hamster in the lower drawers of the wardrobe. When the higher drawer is opened, the cat's meow is heard. When the second drawer is opened, the hamster begins to play.

Design result 2

'A wardrobe shaped like a cat'

The wardrobe can move like a cat. The person orders the hamster to bring a suit. The hamster goes to touch the cat's tail, and then the cat delivers the suit.

Design result 3

'Traveling bag that cares for the pet during travel'

A panel is attached on the side, and an image of the pet is displayed when the panel is opened. Some buttons on the panel enable food to be given to the pet or the pet to be fondled.

Design result 4

'Chest of drawers-ball'

This chest of drawers is ball-shaped and it moves about restlessly. It can enter narrow spaces. Because it is a ball that moves about freely, the chest of drawers can be repositioned easily.

Design result 5

'Desk-chair'

This chair is like a desk. In a word, it is the size of a desk although its appearance is that of a chair. We use it as a chair.

Design result 6

'Chair that can be folded like an umbrella'

A chair that can be folded by the mechanism of a folding umbrella can be stored in a narrow space. It is possible to store it in an underground compartment after use.

Design result 7

'Chair which runs about trying to escape'

This chair runs away when a desk approaches. It resembles a rat being chased by a cat.

Design result 8

'A revolving shoebox'

This rotary shoebox is doughnut-shaped and the size of a person. It rotates when the user stands in front of it, and shoes can be chosen. It is easy to choose shoes appropriate for the outfit because the section for the feet is transparent.

- Task B: Design new furniture starting from the word "Cat-fish"

Design result 9

'A sideboard with a monitor'

Usually an image of fish in an aquarium is displayed on the monitor. However, it is also a television that can be operated by remote control. The monitor is at eye level when the viewer is sitting on a chair.

Design result 10

'A case for marine sports'

It has a heater so items such as a wet suit can be dried. Part of the case is a water tank in which fish can be kept.

Design result 11

'Water tank with casters'

There are legs like those of a chair attached to the bottom of the water tank. Because they have casters, it is possible to move the tank easily.

Design result 12

'A coat hanger that refuses to hang clothes'

This coat hanger will not hang clothes. The clothes will be dropped when hung on this hanger.

Design result 13

'Chest of drawers that eats oneself'

This is a nested chest of drawers behind a door. There are more drawers inside the drawers.

Design result 14

'Water tank table'

This is a table of a hollow structure made of glass. It is possible to store water inside it like a water tank. A fish appears to be swimming in the table.

Design result 15

'Sea cushion'

This cushion can float in the sea. It is possible to sit and to sleep on it. It is possible to join many of them to form a lounger.

Figure 5 shows samples of sketches for design idea No.15 by a subject who is a postgraduate design student.

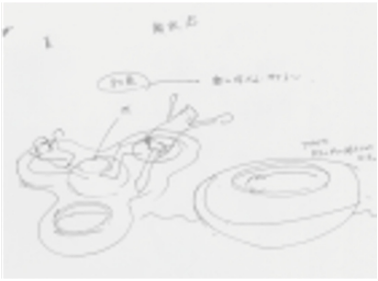


Figure 5. Sketch of design idea No.15 'sea cushion'.

3.6. CREATIVITY EVALUATION OF DESIGN RESULT

The design results (design concepts) are evaluated based on the two viewpoints of practicality and originality; on a five-point scale by 8 people (4 of them are experienced in design).

According to the judging standard, the practicality ratings of No.1, No.2, No.4, No.7, No.12 and No.13 are less than 3 points, whereas the following nine satisfy the required practicality score.

No. 3 'A travelling bag that cares for the pets in transit'
No. 5 'A desk-chair'
No. 6 'A chair that can be folded like an umbrella'
No. 7 'A chair that runs about trying to escape'
No. 8 'A revolving shoebox'
No. 9 'A sideboard with a monitor'
No. 10 'A case for marine sports'
No. 11 'A water tank with casters'
No. 14 'A water tank table'
No. 15 'A sea cushion'

These nine ideas can be called as creative design ideas. Table 2 shows the average rating for these nine design concepts which were satisfied the judging standard. These ten ideas can be called as creative design ideas.

TABLE 2. Creativity evaluation of nine selected design concepts.

No.	Task	Practicality	Originality	Order of high creativity
3	A	3.750	2.875	6
5	A	3.000	2.375	8
6	A	4.125	3.875	1
8	A	3.000	3.625	2
9	B	4.250	2.625	7
10	B	3.750	3.500	3
11	B	4.125	2.000	9
14	B	4.250	3.000	4
15	B	4.125	3.000	5

Originality is high in the order of No. 6, 8, 10, 14, 15, 3, 9, 5 and 11. As a result, it can be said that there is no difference in the between the design tasks A and B. It is said that creativity is also high in this order. Therefore the highest creativity is shown by No. 6.

3.7. EXTENSION OF IDEA SPACE

To identify the extension of idea space, new nouns have been extracted from the utterances recorded during the design task and in the interview, by protocol analysis. There are many new nouns in the nine creative design ideas as we determined (No. 6, 8, 10, 14, 15, 3, 9, 15 and 11), as shown in Table 2 (bold-faced type). This result reveals that there is a relationship between the number of new nouns and high creativity. (No. 3, 6, 8, 9, 10, 12, 14 and 15).

TABLE 3. The numbers of new nouns.

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
New nouns	6	5	**12**	5	5	**7**	3	**11**	**13**	**11**	5	**7**	5	**9**	**21**

Next, focusing attention on the distance between concepts, we examine the relationship between the new nouns arising during the experiment and the terms Cat, Hamster, Fish and Furniture. The distance of the new nouns from Cat, Hamster (Fish) and Furniture is measured using the concept dictionary (Concept Dictionary 2005). The scatter charts for No. 6 and No. 11 are shown in Figures 6 and 7. No. 6 shows the highest creativity, and No. 11 has the lowest creativity result. The abscissa indicates the distance from Cat or Hamster. The ordinate indicates the distance from Furniture.

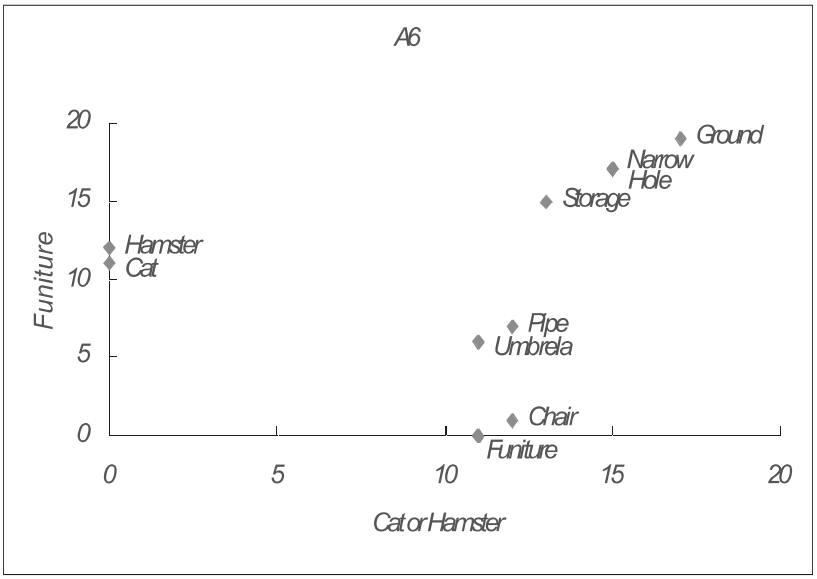


Figure 6. The distance of the new nouns from Cat or Hamster and Furniture in No 6.

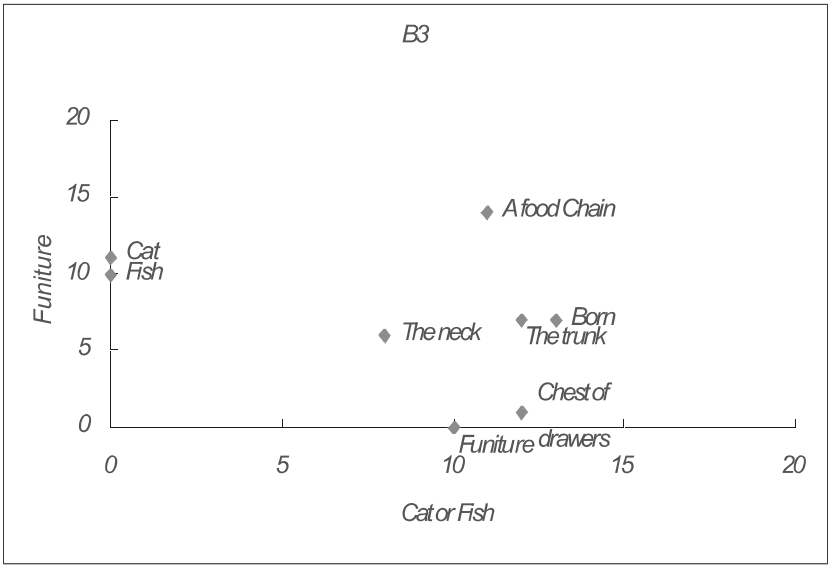


Figure 7. The distance of the new nouns from Cat (or Fish) and Furniture in No. 11.

It is understood from the scatter chart that No. 6 is evaluated to have high creativity since many new nouns are concentrated away from the two axes. We examine the extension of idea space on the basis of distance between the concepts. The extension of the design space (idea space) is defined as follows.

If $n_i(x_i, y_i)$ is a new noun in design result D,

then $\sum_{i=1}^{N} \frac{\sqrt{x_i^2 + y_i^2}}{N}$ (N = number of new nouns) is the extension of idea space.

Table 4 shows the mean and the standard deviation of creativity and the extension of idea space. Figure 8 shows the scatter chart. The correlation coefficient ρ is 0.73087 ($F(1, 7) = 8.02713$, $p < .05$), and it is significant. It is understood that there is a strong correlation between creativity and the extension of idea space. It is shown that there is a strong correlation between the extension of design space and design results with high creativity.

TABLE 4. The standard deviation of creativity and the extension of idea space.

	Creativity	Extension of idea space
$\bar{X}$	2.986	13.862
SD	0.573	1.148

3.8. FACTORS FOR THE EXTENSION OF IDEA SPACE

From the macroscopic perspective, we determined that the design process involves thematic integration or taxonomical blending. We judged the types of relations between the two base concepts ('cat and hamster' or 'cat and fish') during not only the initial process but also the whole process. Table 5 shows extracted relations between the two base concepts and which kind of relation; taxonomical or thematic, stimulated design synthesizing processes in the nine creative design ideas.

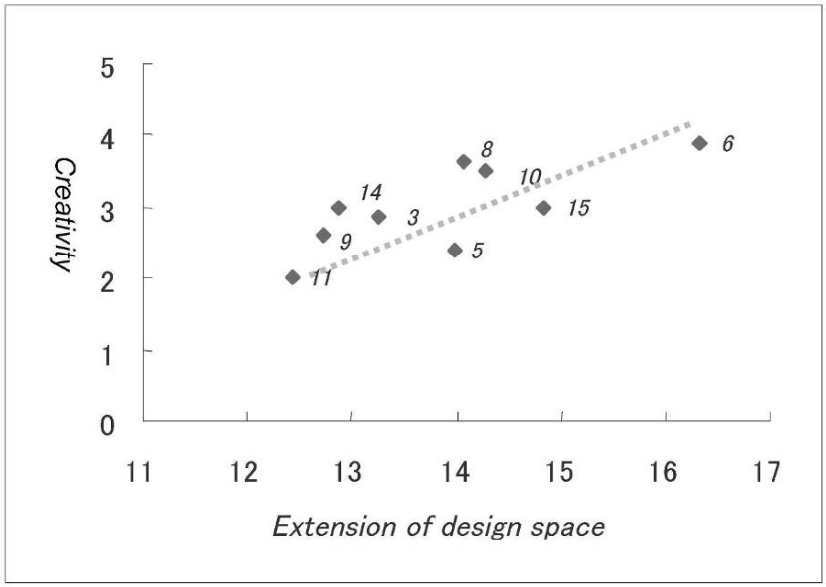


Figure 8. Correlation between creativity and the extension of idea space. As shown above, most of the design results evaluated to have high creativity showed thematic integration from a macroscopic viewpoint.

It is also important to recognize the factors of the extension of design space precisely. Therefore, the design process associated with thematic relations or taxonomical relations is determined from the microscopic perspective. To identify which relations between concepts were connected

with the extension of design space, we examined the concepts (nouns) uttered by the subjects during the design task in detail and judged whether relations were thematic or taxonomical for No.#6 (highest creativity) and No.#11 (lowest creativity) using EDR. Relationships between a pair of two sequential concepts were examined as to whether they were aligned in taxonomical relations or nonaligned but related in thematic relations recalling a scene. The examples of the judgments of design processes are shown in Table 6.

TABLE 5. Types of relations in the top nine creative designs.

	No.	Relations during the initial process	Main relations in whole process	Type of relation
1	No. 6	Cat eats hamster	Cat chases hamster/Thematic	Thematic
2	No. 8	Hamster is a small cat	Cat=Hamster	None
3	No. 10	Cat eats fish	Fish live in water	Thematic
4	No. 14	Cat eats fish	Fish live in the sea	Thematic
5	No. 15	Cat eats fish	Cat sits on a cushion	Thematic
6	No. 3	Both are pets	Both are in a bag	Taxonomical
7	No. 9	Cat eats fish	Fish live in the sea	Thematic
8	No. 5	Cat eats hamster	Desk eats chair	Thematic
9	No. 11	A scaly cat	Cat/Fish has legs	Taxonomical

As a result, a considerable degree of thematic relation was evident in the design process in the case of No. 6, which was evaluated as having the highest creativity, from a macroscopic perspective.

It is understood that the nouns judged to be far from the two axes, Figure 5, were thought up when the subject recollected various scenes. It is thought that the new nouns leading to the extension of the idea space were uttered under the influence of the relationship between cat or fish and the new concept that the subject conceived in the design process.

We extract the characteristics of the factors to extend the idea space, focusing on the thought process during the design task. Therefore, the process for No. 6 shows the highest creativity result and the highest extension of idea space in Figure 7. No.11 shows the lowest creativity among the nine creative ideas and its idea space is only slightly expanded.

Table 7 shows the degree of thematic relations in each process (31% for No. 6 and 16.3 % for No. 11).

TABLE 6. Examples of judgements of design process for No. 6.

Number of sequential pairs	Nouns	Distance from a noun before	Type of relations	Scenes (from the subjects' explanation)
33	Structure – Umbrella	9	Thematic	Structure of umbrella
34	Umbrella– Folding umbrella	1	Taxonomical	A kind of umbrella
35	Hole– Ground	19	Thematic	A hole in the ground
36	Ground –Narrow space	6	Thematic	digging a small hole in the ground
37	Narrow space – Umbrella	17	Thematic	An umbrella which goes into a gap
38	Chair –Umbrella	7	-	Chair is umbrella
39	Folding umbrella – Ground	7	Thematic	Producing a Folding umbrella from the ground

TABLE 7. Features of the process and design space with high creativity.

	No. 6	No. 11
Creativity	3.875 (highest)	2 (lowest)
Extension of design space	16.32 (highest)	12.43 (lowest)
The number of concepts	42	37
Types of relations in initial/whole process	Thematic	Taxonomical
Thematic relation between consecutive concepts	13 pairs (31.0%)	6 pairs (16.3%)

The results indicate that there can be correlations between the thematic relations in the new nouns in terms of time (before and after) and the extension of the idea space in creative design. The result reveals that the thematic relation, which is the principle of concept integration (the 3rd

design process primitive) in the design process, as we described above, may stimulate higher creativity in design through the extension of idea space.

We summarized the results as follows.

(1) The mechanism of the extension of design space, which is associated with design creativity, was confirmed precisely.
(2) From the macroscopic perspective, during the design process associated with higher creativity, conceptual synthesis was initialized by thematic relations between two concepts, and thematic integration took precedence throughout the whole process.
(3) From the microscopic perspective, a characteristic of a design process with high creativity was a high level of thematic relations between the two consecutive concepts.

5. Conclusion

In this study, two topics were examined. First, primitives and principles of the concept-synthesizing process (combining, blending, and integrating) from the viewpoint of creativity were formed. The 1st primitive of the concept-synthesizing process is 'concept abstraction,' and its principle is 'similarity' in 'taxonomical relations'. The 2nd primitive of the concept-synthesizing process is 'concept blending,' and its principle is 'similarity' and 'dissimilarity' in 'taxonomical relations'. The 3rd primitive of the concept-synthesizing process is 'concept integration,' and its principle is 'thematic relations'. Second, the relationships between creativity and the design primitive processes, focusing particularly on the extension process of idea space in terms of the difference between taxonomical relation and thematic relation, were empirically studied. From the results, it was found that, as a consequence of systematizing the concept synthesizing processes during design creation, concept integration (the 3rd design process primitive) may have an effect on higher creativity. Based on the analysis of design space and focusing on the 'thematic relation' between two concepts, the concept-integration process can be associated with the extension of design space.

In this study, we showed 3 primitives. However, there may be other primitives. For example, we hypothesize that the process by which idea space (design space) is created may be another principle. We will continue to describe the forming of primitives and principles in the future.

References

Boden, AM: 2004, *The Creative Mind: Myths and Mechanisms*, Routledge.

Concept Dictionary: 2005, *EDR Electronic Dictionary*, National Institute of Information and Communications Technology, *CPD-V030*.

Ericsson, K and Simon, HA: 1984, *Protocol Analysis*, MIT Press, Cambridge, MA.

Fauconnier, G: 1994, *Mental Spaces*, Cambridge University Press, UK.

Fauconnier, G and Turner, M: 2002, *The Way We Think - Conceptual Blending and the Mind's Hidden Complexities*, Basic Book, NY.

Finke, R, Ward, T and Smith, S: 1992, *Creative Cognition: Theory, Research, and Applications*, Cambridge, the MIT Press, A Bradford Book, London.

Gardenfors, P: 2000, *Conceptual Space*, MIT Press, A Bradford Book, London.

Gero, JS and Maher, ML: 1991, Mutation and analogy to support creativity in computer-aided design, *in* GN Schmitt (ed), *CAAD Futures*, ETH, Zurich, pp. 241-249.

Goldschmidt, G: 2001, Visual analogy, *in* C Eastman, M McCracken and W Newsletter (eds), *Cognition in Design Education*, Elsevier, UK, pp. 199-219.

Harakawa, J, Nagai, Y and Taura, T: 2005, Conceptual synthesis in design creation, *Proceedings of the 1st IDC*, in CD-ROM, 2005.

Hayashi, M: 2002, *Three Swedish Designs*, Living Design, 21, Living Design Center, Tokyo, pp. 76-82.

Living Design Club (ed): *Living Design Collection,* (Jacobsen, A: Swan Chair 1958, Tendo: Mushroom-stool 2003, Long Island : Easy Chair 2002), Available Online: http://www.ozone.co.jp, last accessed November 2005.

Lawson, B: 1997, *How Designers Think*, Architectural Press, Oxford.

Taura, T, Nagai, Y and Tanaka, T: 2005, Design space blending-A key for creative design, *Proceedings of International Conference on Engineering Design*, the Design Society, Melbourne, CD-Rom.

Tuara, T, Yoshimi T and Ikai, T: 2002, Study of gazing points in design situation- A proposal and practice of an analytical method based on the explanation of design- activities, *Design Studies* **23**(2): 165-186.

Taura, T and Nagai, Y: 2005, Primitives and principles of synthetic process for creative design,*in* JS Gero and ML Maher (eds), *Proceedings of Computational and Cognitive Models of Creative Design VI*, pp. 177-194.

Wisniewski, EJ and Bassok, M: 1999, What makes a man similar to a tie?, *Cognitive Psychology* **l**(39): 208-238.

ROBUSTNESS IN CONCEPTUAL DESIGNING: FORMAL CRITERIA

KENNETH A SHELTON
Aerospace Corporation, USA

and

TOMASZ ARCISZEWSKI
George Mason University, USA

Abstract. The purpose of this paper is to propose formal robustness criteria and a visualization concept for these criteria to be used in conceptual designing. The criteria, called the "Component Distance" and "Value Distance," when added together form the "Configuration Distance." The developed visualization concept, called "Design Solution Topography," allows visualization of a population of design concepts in terms of their performance and their component and value distances. The proposed criteria address three sources of variance in engineering systems, namely manufacturing error, system degradation and parts availability. When incorporated into an evolutionary conceptual designing method, being developed by the first author, the criteria will enable designers to evaluate and ensure robustness in selected designs that also satisfy the desired performance objectives at a minimum threshold or greater.

1. Robustness In Conceptual Designing

Physical systems are the primary means by which mankind enables itself to achieve a set of desired capabilities and performance characteristics in the physical world. While performing their functions over their operating lifetime, it is desirable that they do so in the presence of variances in configuration and environment. If they do, then they have a feature called robustness. If not, then they are brittle and prone to failure. This paper examines the feature called ìrobustness" and proposes formal robustness criteria for use in the evolutionary designing of complex engineering systems. The reported results are initial products of a research project on an evolutionary design method for robustness being developed by the first author at George Mason University.

J.S. Gero (ed.), Design Computing and Cognition '06, 461–479.
© 2006 *Springer. Printed in the Netherlands.*

Robustness is an important issue in conceptual designing. There are three ways in which ìrobust" is used in conceptual designing: (1) a robust method of conceptual designing that can be applied to different types of problems, (2) a robust design concept that can be used in different application types, and (3) a robust design concept that tolerates variances from specifications that are caused by various sources. This paper deals with the third case.

In general, designs that are performance-sensitive to relatively small changes in component attribute values are ill-suited to application and implementation outside of laboratory conditions. This is particularly the case in complex systems such as satellites and aircraft. In such systems, due to uncertainties and variances, robustness can be more important than optimizing performance against the objectives set. Although there is a compelling need for robustness in designs, it has proven difficult to achieve in practice. History abounds with designs that failed under unexpected or unplanned conditions - the Challenger Space Shuttle, 1996 and 2003 power grid crashes, London Millennium Bridge, and the 1940 Tacoma Narrows Bridge. These engineering design failures all contain a common thread - when put in implementation, designs that were thought to be acceptable exhibited catastrophic performance degradation under conditions that were not foreseen during the design development process. Extreme sensitivity to changes in a subset of the overall environmental or configuration parameters led to a complete breakdown of the system integrity.

Crow proposed that robustness could be achieved through two approaches: brute force of adding design margin and tightening tolerances; or through logical, methodical, ëintelligentí designing that produces the design concepts to be inherently robust. Here ëintelligentí doesnít necessarily mean computer-driven - it simply means that robustness as a feature is intentionally integrated at the outset, and a formal means to ensure it in the resulting design concepts is used throughout the design development. He also provided a conceptual construct for robustness that can be modified slightly by adding variation resulting from changes in parts availability as a fourth characteristic. This revised construct for robustness is what will be used in this paper - robustness is a function of variations resulting from: (1) Manufacturing Errors, (2) System Degradation, (3) Operating Environment, and (4) Parts Availability.

From a practical standpoint, robustness is considered sufficient if the performance of the design is within some pre-defined acceptable limits for relatively reasonable changes in either the design configuration, design specifications or the operating environment. How large these changes are in absolute terms is a function of the problem domain and operating environment. Therefore, during the design development process, designers are required to establish the thresholds to which variance must at least be considered, with the goal to go beyond that if possible, realizing there will

be a limit to the variances that can be accounted for because of either physical limitations or cost impact.

Robustness is often in the undesirable situation of competing against the maximization of performance against a set of objectives. Designers typically have the primary goal of maximizing performance. However, while optimal performance is pursued, it is often difficult to determine if the resulting designs are robust. It is also often the case that robustness and optimal performance are not in harmony - what is the best approach for robustness does not result in optimal performance and vice versa. To get truly optimal performance, designs may be required to be in very specific configurations and operating environments for a large percentage of the time or incur severe performance degradations. However, designs that are very specification-sensitive are difficult to manufacture and operate. To make matters worse, this situation may not be intuitively obvious to the designer as the existing conceptual designing methods are not conducive to providing this insight. Another problem is that oftentimes designers have little appreciation for or understanding of robustness.

To that end, it is desirable to use a designing approach that explicitly provides insight to and makes quantitative or qualitative assessments about robustness within the resulting conceptual designs. This insight should provide for the sources of variance noted earlier. In this paper, the four sources of variance will be handled in the following manner. First, manufacturing errors and system degradation represent changes in attribute values. Thus, both can be quantified and assessed as variance in the value of attributes in the components that comprise the design concept. Secondly, parts availability represents changes in component types. Thus, as component types are added or subtracted from the original design concept to generate new design concepts, parts availability can be quantified and assessed. Thirdly, operating environment variance has not been studied in this effort. Researching the limitations of first principle knowledge and modeling sufficiency is beyond the scope of this effort. For this effort, only existing modeling and simulation tools will be considered.

2. Robustness in Conceptual Designing

Many different conceptual designing approaches, such as human-based trial and error, heuristic, and random approaches, to formal approaches like Axiomatic Design Theory, Pareto Optimal models and Evolutionary Computation. In all these approaches, robustness has been a difficult feature to implement or to even provide a method to assess its qualities in candidate design concepts.

Various researchers have offered approaches tailored to specific conceptual designing methodologies. The Taguchi Method forms the

historical basis for robustness studies, which is a statistical approach that formulates manufacturing variances as loss functions. Batson and Elam surveyed robust design approaches and developed a set of basic tenets, primarily focused on ìDesign for Reliability". Hu et al. (2001a, 2001b) and Melvin and Deo (2002) noted the limitations of the Taguchi Method, namely large data sets are needed and a lack of application to the conceptual designing phase. The authors offered processes that use a combination of traditional Axiomatic Design and TRIZ (Hu et al. 2001a), and Axiomatic Design alone (Melvin 2002), that formatted the noise sources / loss functions as objective functions. Chen et al. (1999) used Design Capability Indices to predict design performance variance as a robustness metric. Chen, Biggers et al. (2000) examined a robust design approach using their Robust Concept Exploration Method. The Six Sigma Method focuses on quality control and quality improvement. The final output, if correctly implemented, is a more robust and fault-tolerance product. Chen, Sahai et al. (2000) and Allison et al. (2005) and other authors have offered multi-objective programming methods that represent the design as a set of simultaneous equations subject to max/ min objective functions.

This research effort proposes using an Evolutionary Computing approach to enable the development of robust solutions in the conceptual designing phase. In this approach, conceptual designs are generated using the methods developed in Evolutionary Computing. Evolutionary design development has been extensively studied and developed over many years by authors such Axelrod (2005), Kicinger (2005), Potter (1997) and Wiegand et al. (2001).

3. Assumptions

Two robustness concepts are proposed in this paper, called the "Configuration Distance" and the "Design Solution Topography." They are described in detail in Section 5. To develop and implement these concepts, several key assumptions about the problem definition were made. They are that (1) design concepts can be represented in a component-allele structure, (2) they are collections of components, (3) each component is described by attributes, (4) each attribute is represented by one or more alleles, the values of which define the value of the attribute, and they have value structures that best fit the problem definition (integer, real, binary, etc.), (5) the attribute alleles can be grouped in strings that comprise a genome that uniquely describes a component in terms of its attributes and their values, (6) the component genomes can be composed in groups to form conceptual designs, and (7) objectives can be represented in terms of the component allele values. Essentially, the design problem can be formulated in an evolutionary computing context. If the problem is not conducive to this, then these concepts will not be applicable to that conceptual designing task.

4. Relevant Concepts

The proposed robustness criteria have been developed using three known concepts called "Landscape Theory" (also known as the "Fitness Landscape"), the "Hamming Distance," and the "Spatially Embedded Solution Space."

4.1. LANDSCAPE THEORY AND FITNESS LANDSCAPE

Axelrod "developed the ìLandscape Theory" where an n-dimensional plot is used to show the solution landscape for a feature or attribute that is a function of the axes parameters. This gives the ability to graphically represent complex relationships in a visually intuitive manner, and also allows identification of maxima and minima that can be difficult to infer from the raw equations and input data. Weigand et al. (2001), Stadler and Stephens (2002) and Popovici and DeJong (2002) developed an extension of the landscape concept called the "Fitness Landscape". A fitness landscape is a plot that shows the design performance for an objective function given a specific configuration of attribute values, which are plotted on the other axes. Theoretically, this can be an n-dimensional surface. Visually, it is usually presented in two attributes with the third axis being the objective function performance. This is shown in Figure 1.

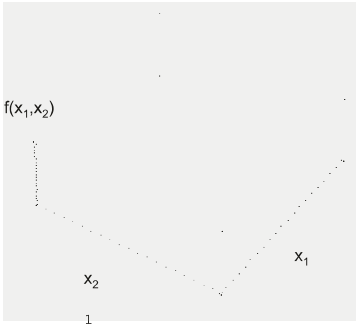


Figure 1. Example of a continuous fitness landscape.

In this research, the Fitness Landscape will be modified such that the axes (X1 and X2 in the Figure) represent variances in the design concept as a whole instead of variances in a single attribute within it. Furthermore, the points plotted on the landscape will represent solutions for different design concepts in the population vice the same design concept with a different attribute value in a traditional fitness landscape. This will allow the designer to make assessments about the relative quality of the design concept compared to other members of the same population. To do this enhancement to the fitness landscape concept, two other concepts called the

"Hamming Distance" and the "Spatially Embedded Solution Space" will be modified and integrated with the "Fitness Landscape" into a new the method.

4.2. HAMMING DISTANCE

The Hamming Distance is a standard measure of the number of changes that are required to transform one string into another string of equal length, and is used in information string problems. It is calculated in the following manner: suppose String #1 = [BECAME] and String #2 = [BECALM], then the Hamming Distance = 2 as the two letters ME must change to LM.

It has been examined by others, such as Werfel et al. (2002), in determining diversity in collections of rules and population members. In general, it does not have a direct correlation to performance - i.e. calculating the Hamming Distance does not necessarily provide any insight into variance in performance for those two members. The Hamming Distance is a metric - it describes characteristics and features of the data, but does not contain information about the data itself. For this research effort, a measure that is similar to the Hamming Distance will be used, but it will have features more useful for the designer.

4.3. SPATIALLY EMBEDDED SOLUTION SPACE

Paigie and Mitchell (2002) developed the idea of a Spatially Embedded Solution Space. Here, the individual members of a population are distributed on a grid, and members are controlled and managed based on the spatial distance between them. This management can be of several types, such as controlling the interactions between population members in an evolutionary process. The population would be distributed on the grid and breeding actions, such as crossover, would be regulated based on the spatial separation between the members. Members located in close proximity would be allowed to breed with each other at a much higher rate than those with greater geographical separation. Distribution on the grid may be determined by any number of methods, to include a simple random distribution.

This research effort modifies the concept to define spatial separation to be proportional to variance in design configurations. Thus, large distances indicate little commonality, while small distances indicate a high degree of commonality. These distances are calculated using a proposed concept called the "Configuration Distance". This can be visualised in Figure 2. In this example, taking population member DD, the population members located one space away spatially have high commonality with DD. Similarly, members located around the perimeter have low commonality with DD. Variance is proportional to the spatial separation between two

locations. In principal, this distribution can be extended out in either axis direction as far as required to account for all types of variance that are desired to be displayed, evaluated and managed. The spatially embedded system thus provides a mechanism to illustrate and manage collaboration and variance between members of a population.

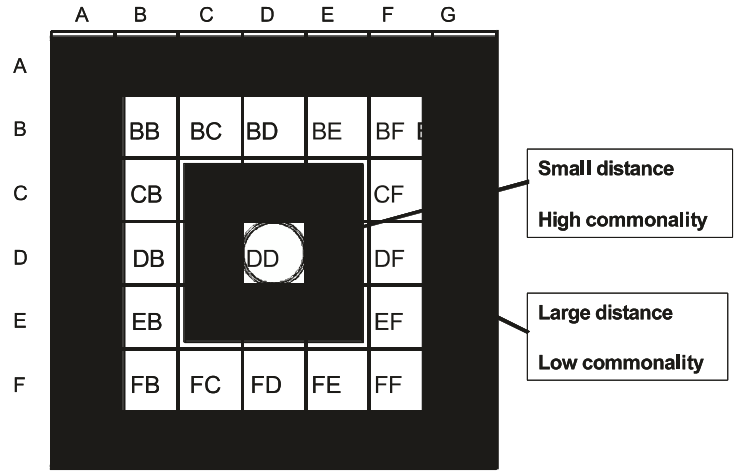


Figure 2. Spatially Embedded Solution Space.

5. Robustness Criteria and their Visualisation

In this research effort, the robustness of a design concept is based on how the variance between design concepts impacts performance. To analyse such relationships, robustness criteria and a visualisation concept have been developed that will be implemented in an evolutionary designing method. The criteria are called "Value Distance" and "Component Distance." Their sum is called the "Configuration Distance." The proposed visualisation concept is called the "Design Solution Topography." It allows visualisation of a population of design concepts in terms of their performance as well as of their value and component distances.

5.1. ROBUSTNESS CRITERIA – CONFIGURATION DISTANCE

The Configuration Distance provides the ability to manage and assess robustness of the design concepts considering both the qualitative (structural) and quantitative (parametric) differences between any two concepts. In the same vein as the Hamming Distance, the Configuration Distance is a measure of the variance between two design concepts. It is comprised of two parts – the Value Distance and the Component Distance, and the formula used is: Configuration Distance = Value Distance + Component Distance = (attribute variances) + (component type variances).

A design concept is herein understood as a collection of components of various types and configurations (subsystems). The components contribute to one or more performance objective for the system. The variance between two design concepts can therefore be taken in two parts, (1) variance in components of the same type that are common to both design concepts (quantitative or parametric differences), and (2) variance from component types that are not common to both design concepts (qualitative or structural differences). The first kind of variance is herein referred to as Value Distance, and the second is herein referred to as Component Distance. The Configuration Distance is the sum of the Value Distance and the Component Distance. To illustrate, suppose two design concepts, A and B, can contain four different component types, numbered 1-4, which are described by strings of alleles whose values are either 0 if the component is not present, or of an integer value on the interval [1,25] if the component is present. The component types have the form of: Component type 1 has 3 alleles, [a11, a12, a13]; Component type 2 has 4 alleles, [a21, a22, a23, a24]; Component type 3 has 3 alleles, [a31,a32,a33]; and Component type 4 has 5 alleles, [a41,a42,a43,a44,a45]. Design concept A contains component type 1 of alleles [2,4,6] and type 2 of [8,10,12,14]. Design concept B contains component type 1 of alleles [17,19,21] and type 3 of [7,9,11]. Value Distance: The Value Distance is defined as the attribute variance between the two designs for the parts that are comprised of the same component types. From the example, Design Concepts A and B have one component in common – component type 1. Thus, the Value Distance between A and B can be calculated as |2-17| + |4-19| + |6-21| = 45. Component Distance: The Component Distance is defined as the maximum variance from each component that is not in common between the two design concepts. Design Concept A contains component 2 uniquely, and Design Concept B has component 3 uniquely. The Component Distance is equal to the attribute values for each component vs the null set: |8-0| + |10-0| + |12-0| + |14-0| = 44, and |7-0| + |8-0| + |9-0| = 24. Component Distance = 44 + 24 = 68. Configuration Distance is the sum of the Value Distance and the Component Distance. Therefore, Configuration Distance = 45 + 68 = 113. The value of the Configuration Distance, like the Hamming Distance, is a metric – it is not something intrinsic that could describe a feature of the design concepts themselves. It represents an assessment of variance between two designs. If it is calculated that Design Concept A and a third Design Concept C have a Configuration Distance of 50, then it can be said that Design Concept #3 has less variance from Design Concept A than does Design Concept B. The Configuration Distance allows for the handling of complex, interdependent changes without the need to track and analyze these relationships.

5.2. CRITERIA VISUALISATION - DESIGN SOLUTION TOPOGRAPHY

The "Design Solution Topography" concept is based on the two concepts of the Spatially Embedded Solution Space and the Fitness Landscape. In this case, the Fitness Landscape is modified so that instead of laying out the two baseline horizontal axes as attributes, one axis represents variance in the composition of component types in the design concepts - variance in Component Distance. The second axis represents the population members that vary in attribute values, but have the same composition of component types - variance in Value Distance. As before, the third axis is performance for a given design concept considering a particular objective function. In this way, each coordinate indirectly represents an entire design concept instead of only a pair of attributes within a single design concept as is the case in a standard fitness landscape. In effect, this approach transforms the N-1 dimensions of the generalized fitness landscape (N-1 being the total number of discrete attributes that together define the performance function plotted as the Nth axis) into a 2-axis representation of the total overall design concept considering its qualitative and quantitative characteristics, Figure 3.

The design concept at location (0,0,X) is termed the "base design" (reference point) from which all Configuration Distance measurements are made as, by definition, the Value and Component Distances for the base design from itself are zero (where X = performance). This visualisation enables the evaluation of the robustness of any population member in the topography regardless of its configuration or the order in which it was created by simply redrawing the topography with that selected design concept as the base design at (0,0,X) and distributing the rest of the population according to the Configuration Distance from that design concept.

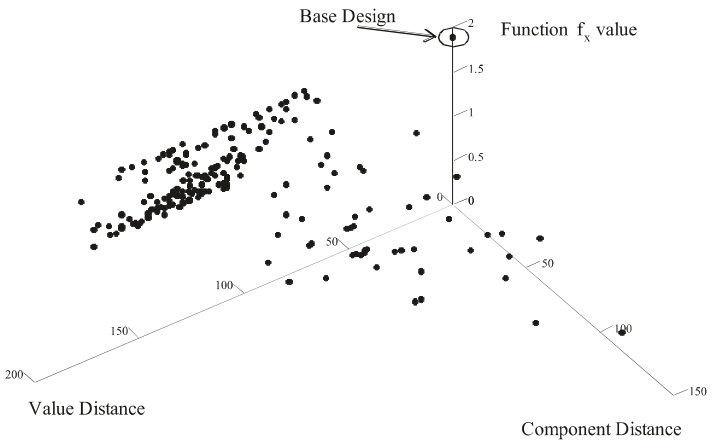


Figure 3. Example design solution topography.

Because the design concepts being considered are generated using evolutionary methods, each instance of the Design Solution Topography contains the landscape of a population of design concepts for a given generation. This topography is used to select the best performers in terms of robustness, and those selected are the seed population for the next generation. Evolutionary processes are then conducted to produce the next generationís population, and then a new Design Solution Topography is made. Figure 3 illustrates an entire population of design concepts. By distributing the members on the Design Solution Topography based on Configuration Distance, and by evaluating the robustness of those design concepts in a way that is proportional to it, a merger of the Spatially Embedded Solution Space and the Fitness Landscape concepts is achieved by making commonality and population management based on spatial separation.

There are two important features of the Design Solution Topography. First, the design concepts themselves, represented by the individual points, are located in a way that describes their characteristics. From Figure 3, the points distributed on the plane of Component Distance = 0 have the same composition of component types as the base design and vary only in attribute values (Value Distance). Additionally, vertical slices where Value Distance equals a specific amount show the design concepts with the same aggregate variance from the base design. Note that two designs can have the same Value Distance from the base design but have different performance values. This is based on the discrete variance sources being different for the two, but the aggregate variance being the same. In this way, the Design Solution Topography allows an evaluation of robustness to design concept level tolerances when the variance results from different discrete sources. Thus, variance can be caused by multiple attribute changes, but the topography allows it to be visualised and evaluated in the aggregate. Similarly, two design concepts with the same component distance show that they each have some number of component types that differ from those of the base design, and that the aggregate variance is the same.

The second important feature is the topography distribution trend. If the data points are grouped in a tight cluster, then it indicates that the diversity in the population is poor and the evolutionary process may not be considering a sufficiently broad sampling of possible solutions. Additionally, as the evolutionary process reaches later generations, the overall performance and robustness quality should have a positive trend and plateau at convergence.

5.3. ROBUSTNESS EVALUATION USING THE CRITERIA

As noted earlier, variance that can be accepted during the useful life of the system from the three sources being considered (manufacturing errors, system degradation and parts availability) generally cannot be infinite in amount - there is some reasonable limit to the amount of variance that the physical system can be expected to accommodate. This limit is represented by the tolerances that are assigned to the components and their attributes. Comparing these tolerances to the Configuration Distance values in both the Component and Value Distance allows the designer to determine if the variance is within the aggregate tolerances. Robustness, then, is the ability to obtain design concepts whose performance does not degrade unacceptably given variance up to, and preferably greater than, those aggregate tolerances. In evaluating robustness, greater consideration is given to the cases where the variance is within the tolerances.

Lesser consideration is given to situations where the Configuration Distance exceeds the tolerances, such that the configuration is outside the pre-established expected tolerances. It would be undesirable to discard an otherwise high-quality design concept based on substandard out-of-tolerance characteristics, however, it is desirable to recognise those designs that perform well in both conditions. Therefore, if a design concept shows satisfactory robustness and performance beyond the expected levels of variance, then it receives consideration for this. Conversely, if robustness or performance drastically collapse, then this information is also very useful and is factored in to more accurately assess the overall quality of the design concept. This is because it is very difficult to accurately predict all possible operating situations (as discussed in section 1), therefore a design concept that fails catastrophically in this area contains a greater risk of operating ailure than design concepts that maintain robustness outside the anticipated tolerances. Ideally, a final chosen design concept would not only have robustness and performance within the expected operating profile, but would continue to do so for a substantial envelope beyond that expected profile as well in order to accommodate unplanned contingencies. This proposed method has the capability to evaluate this and conduct population management with that goal in mind.

Thus, in the aggregate, the Design Solution Topography allows for an assessment of robustness from both perspectives - the Value Distance and the Component Distance. It merges the concepts of the Spatially Embedded Solution Space and the Fitness Landscape to provide a visually intuitive and informative display that allows a simultaneous evaluation of robustness and acceptable performance. Also, it provides a logical and methodological foundation for conducting population management within the evolutionary processes that generate the various design concept populations.

5.4. PROCEDURE AND EXAMPLE PROBLEM

The proposed procedure to implement the Configuration Distance and Design Solution Topography methods has six parts, namely: (1) Establish the Problem Representation Space, (2) Generate the Initial Population, (3) Evaluate the New Population for Robustness and Performance (4) Rank Order the Population for Robustness and Performance and Conduct Population Management, (5) Detailed Robustness Assessment, (6) Generate the New Population, and (7) Iterate Back to Step 3 Until Convergence. This is illustrated in Figure 4.

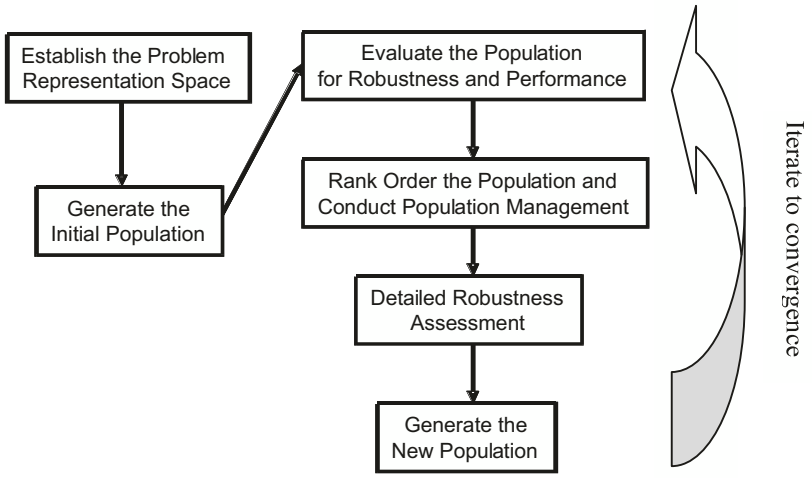


Figure 4. Design procedure.

The first step establishes the problem representation space, comprised of the working population size, format for the design concepts and component types, format of the objective(s), including their relative weights, and the simulation or modeling tools used to evaluate performance. The second step initialises the evolutionary process with a starting population. This population can be developed using a number of approaches, such as random assignment of components and attributes, using existing patented solutions, or previously developed candidate solutions. Regardless of the approach, the initial population should have sufficient diversity to allow a faster sampling of the spectrum of potential design concept configurations. If a large percentage of the initial population is identical or nearly so, then the evolutionary process could become trapped in local optima.

Next, the population is assessed based on performance and robustness for the objective function(s). Performance assessment is simply an evaluation of the performance of the design concept. Robustness is assessed by calculating the amount of change in performance to the Configuration Distance between each design and the rest of the population members, each

member in turn in the same one-on-one comparison. In this way, the Configuration Distance and Design Solution Topography are used as an initial broad filter to determine robustness. The desirable condition is that the change in performance per unit of Configuration Distance is minimized, or alternatively the Configuration Distance per unit change in performance is maximized. Greater consideration is given to relatively smaller Configuration Distances than large ones as these are closer to the maximum expected tolerances.

The rank order and population management step puts the population members in rank based on performance and robustness. The population is then culled in two steps based on the aggregate assessments of performance and robustness. The amount of culling can be tailored to the problem. For this example, the bottom ° of the population in terms of performance are culled first, and then another ° of the original population size are culled being the lowest in robustness. This leaves Ω of the original population as the reduced members that will compete for the opportunity to produce offspring designs in an evolutionary manner.

Next, a second, more refined, robustness assessment is made to the remaining population members. For each in turn, a number of near neighbors are generated that vary only in Value Distance and such that the variance is within the maximum expected tolerance. A check is then made whether the performance changes by more than the defined maximum allowable amount. If it does, then the designís breeding likelihood is severely penalized. This is because it clearly shows unfavorable robustness, as its performance alters by more than the maximum allowable amount while the variance is within the levels that are expected to be encountered. If the design passes this check, then its breeding proceeds and it generates offspring at a high likelihood compared to the rest of the population.

Lastly, new population members are generated to expand it back to the original size. These new members differ from the original ones in both attribute values and component type configurations. A sufficiently high rate of mutation in both features is incorporated to ensure diversity in the new population from the previous one. In the early stages of execution, the design concept with the best robustness for one generation may be outdone by one or more newly created design concepts in the next generation. At some point, though, the process will converge and newly generated design concepts will be only as good or worse robustness-wise than the previous generationís best design concept(s). When this occurs, the process terminates and the final population is output.

Taking the earlier example, suppose design concepts in a problem can contain a single instance of up to four different component types, numbered 1-4, which are described by strings of alleles whose values are either 0 if the component is not present, or of an integer value on the interval [1,25] if the

component is present. The component types have the form of: Component type 1 has 3 alleles, [a11,a12,a13]; Component type 2 has 4 alleles, [a21,a22,a23,a24]; Component type 3 has 3 alleles, [a31,a32,a33]; and Component type 4 has 5 alleles, [a41,a42,a43,a44,a45]. Suppose one of the objective functions, f 1 = COS [p * (a11 + a21 + 1) / (a11 * a21 + 1)] + COS [p * (a31 + a41 + 1) / (a31 * a41 +1)] + 2 (which results in an objective value on the interval [0,4]). For allele values of [0,25], the Fitness Landscape has a value distribution for each allele pair as shown in Figure 5, where the aggregate for f1 would be a sum of two of these Fitness Landscapes. The locations of a sampling of the allele pairs of the top design concepts in the 25th generation are annotated on Figure 5.

Similar objective functions were formed named f2, f3 and f4, which are functions of the remaining alleles in combinations of two pairs of two alleles. From the example plot, the function has the characteristic of poor robustness at the edges of the distribution. As the values of the allele pairs increase to (25,25), the function enters a plateau region of favorable robustness in that small changes in allele values result in small changes in objective function value.

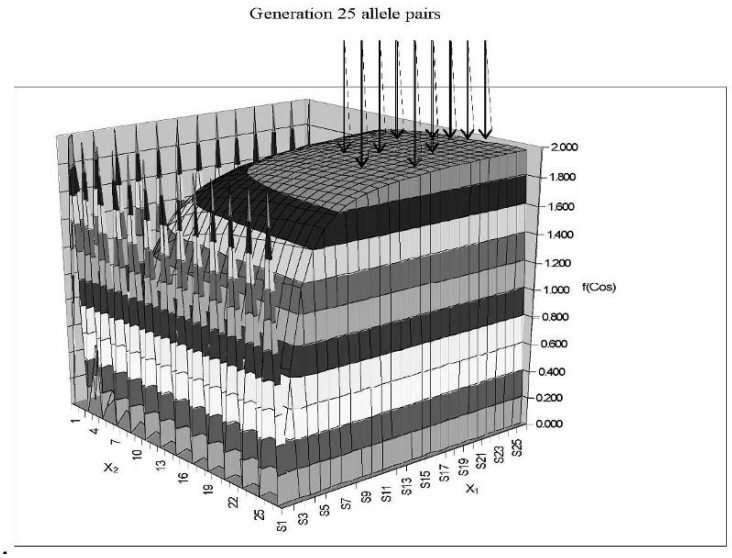


Figure 5. Objective Function Fitness Landscape.

Based on this distribution, the more robust solutions are in the region beyond the rise, starting approximately at the point (14,14) and extending to (25,25). Note that this is not a maximum performance location - that is located along the edge where single points oscillate between 2.0 and 0, resulting in global maximum performance but very unacceptable robustness.

To illustrate the robustness selection process, a performance threshold of 3.8 is chosen so that the performance goal will attempt to drive the results to either the edge points where performance is optimal, or to the region bound on the lower end by the points (10,25), (12,18), (14,15), (18,12), and (25,10), and extending to the apex at (25,25).

If performance is the only goal, then the edge points are clearly the best - however, they are also very unstable. If the method executes as desired, then the robustness feature should discount these performance-optimal cases and instead prefer the choice of the robust solutions in the opposite region. The implementation model shows favorable behavior in examining this problem. Using a randomly generated initial population, Figure 6 summarizes the top results for generations 5 and 25.

5GENERATIONS

Desing#	CompType	All 1	All2	All3	All4	All 5	Compy/n
1	1	25	20	19	0	0	1
	2	22	16	23	18	0	1
	3	16	21	20	0	0	1
	4	22	23	13	17	19	1
2	1	25	20	19	0	0	1
	2		16	22	18	0	1
	3	22	21	20	0	0	1
	4	12	7	12	17	19	1
3	1	25	20	19	0	0	1
	2	18	16	22	18	0	1
	3	16	21	20	0	0	1
	4	12	5	12	17	19	1
4	1	25	13	19	0	0	1
	2	14	23	22	18	0	1
	3	22	21	20	0	0	1
	4	12	23	12	17	13	1

25GENERATIONS

Desing#	CompType	All 1	All2	All3	All4	All 5	Compy/n
1	1	25	20	19	0	0	1
	2	18	16	22		0	1
	3	22	21	20	0	0	1
	4	12	7	12	17	19	1
2	1	25	13	19	0	0	1
	2	14	23	22	18	0	1
	3	22	21	20	0	0	1
	4	12	23	12	17	13	1
3	1	25	13	19	0	0	1
	2	14	21	22	18	0	1
	3	15	22	20	0	0	1
	4	12	16	12	17	15	1
4	1	25	20	19	0	0	1
	2	22	16	23	18	0	1
	3	16	21	20	0	0	1
	4	22	23	13	17	19	1

Figure 6. Example Problem Results for Generations 5, 15 and 25.

The Design Solution Topography is shown for these generations also in Figure 7. These two data products capture the results of the method's algorithm, Figure 7.

The relationship between the Fitness Landscape from Figure 5 and the Design Solution Topography in Figure 7 is more easily visualised in this particular example because there are only two variables. For an N-variable problem, the objective function plot would be much more complex - it would also be N+1 dimensional - while the Design Solution Topography would still be a 3-dimensional transformation of it. Note in Figure 5 that as the generations progress from 5 to 25, the top designs show the allele pairs being selected in the favorably robust region and staying away from the unstable outer margin. Also, the Design Solution Topography has the trend of the top performers generally improving in performance, as the distribution rises on the f1 axis, and begins to cluster closer to the base design axis at (0,0,f1). The Ω of the population that is generated through the evolutionary

methods also continues to show a broad diversity of Value and Component Distances, which ensures that the identified robust solutions represent the best configurations.

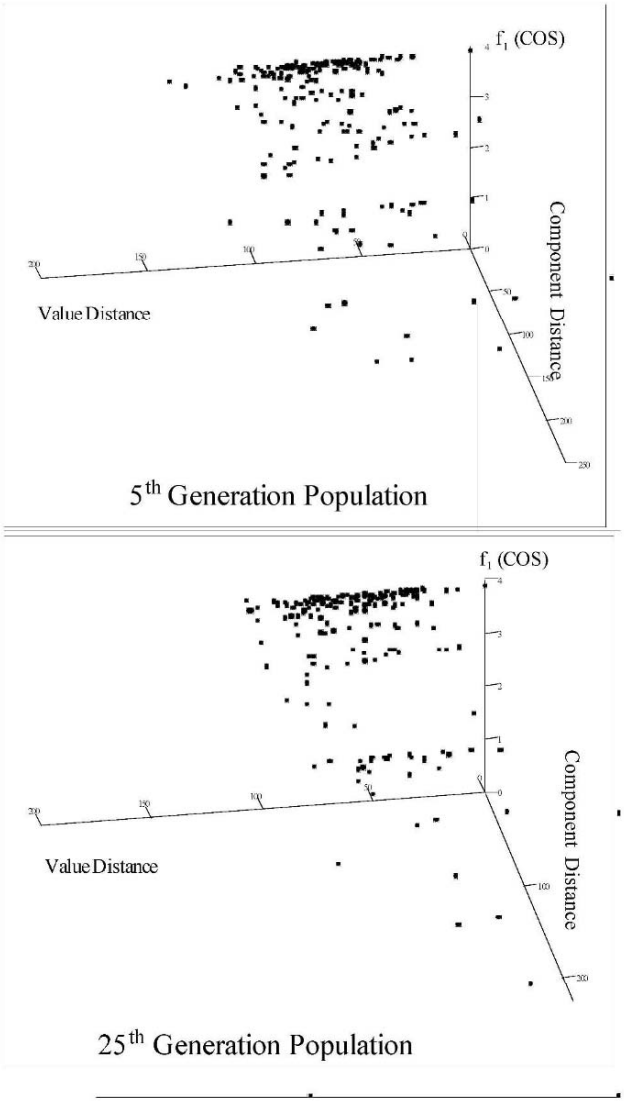


Figure 7. Design Solution Topographies.

6. Conclusions and Areas of Future Research

The proposed criteria are intended to enable designers to focus on robustness as a primary goal of a conceptual designing process. The proposed

visualisation method allows the designer to analyse and interpret the criteria to manage and evaluate the design concept populations. They are offered as an enhancement over traditional methods that focus on optimal performance, and over manually intensive approaches that provide robustness evaluation features. The next step in the research is to demonstrate the utility of the proposed approach in a method that implements the Configuration Distance and Design Solution Topography. This method and its implementation in a computer-based model have been developed as part of a research effort at George Mason University, and a successive paper to this one will be written to provide the results of those efforts.

In addition, there are areas of this research for future efforts. As noted, one of the primary sources of variance is the operating environment. This effort did not examine operating environment variability. Secondly, it is thought that the process of calculating the Configuration Distance and mapping it to the Design Solution Topography could provide insight into the first-order principles of the problem representation space in areas such as attribute dependencies as the concepts allow multiple changes in many attributes and multiple changes in component types. If trends in performance were recorded and analysed based on these variances, then it may be possible to determine interdependencies and interactions between components that are not initially apparent, especially in complex design problems.

Furthermore, this insight could provide the ability to direct problem definition modification by identifying the exact source and nature of contradictions that cause infeasibility. Information on the trends and interdependencies of the various component types and attributes from the Configuration Distance calculations and the Design Solution Topography could provide the capability to map backwards from the results to locate where two or more objectives result in infeasibility. Also, this same mapping process could identify component types that are ill-suited to the problem definition and/or the objectives being desired. Such identification could allow the designer to seek out other alternative component types that provide a better fit to the needs of the design concept - a need that might not be apparent at the outset of the conceptual designing process. If such a mapping process were feasible, then it might also be possible to evolve the problem definition in real-time in addition to the design concepts themselves.

7. Definitions

Engineering design - a complete description of a future engineering system
Design concept - an abstract description of an engineering system that deal with its functionality, outputs, form, and configuration but does not have

detailed component information. It is mostly symbolic, and qualitative attributes are used.

Detailed design - the final specification of a future engineering system in which mostly numerical/quantitative attributes are used, such as dimensions, cross sectional areas, etc.

Attribute - describes a physical characteristic, system behavior or performance of the design. Attributes may be either static or variable in value throughout the design development process.

Component - a fundamental part of the design that may be either a lowest physical decomposition (i.e. board / box-level) or a lowest functional / operational decomposition. Components may be of different types or may be of the same type but with different attribute values.

References

Allison, J, Kokkolaras, M, Zawislak, M and Papalambros, P: 2005, On the use of analytical target cascading and collaborative optimization for complex system design, *6th World Congress on Structural and Multidisciplinary Optimization,* pp. 3091-3100.

Arciszewski, T and DeJong K: 2001, Evolutionary computation in civil engineering: Research frontiers, *Civil and Structural Engineering Computing*: 161-185.

Axelrod, R: 1997, The complexity of cooperation: Agent-based models of competition and collaberation, *Princeton Studies in Complexity*.

Batson, RG and Elam, ME: 2002, Robust design: An experiment-based approach to design for reliability, *Maintenance and Reliability Conference (MARCON).*

Black, PE: 2004, Hamming distance, *National Institute of Standards and Technology (NIST) Dictionary of Algorithms and Data Structures*.

Campbell, MI, Cagan, J, and Kotovsky, K: 2001, Learning from design experience: TODO/ TABOO Guidance, *ASME DETC2001*, DTM-21687.

Chan, K-Y, Skerlos, SJ, and Papalambros, PY: 2005, An adaptive sequential linear programming algorithm for optimal design problems with probabilistic constraints, *ASME 2005 IDETC/CIE*, *DAC 8448*.

Chen, F, Biggers, SB, Fu, W and Latour, RA: 2000, An affordable approach for robust design of thick laminated composite structure, *Optimization and Engineering* **1**(3): 305-322.

Chen, W, Sahai, A, Messac, A, and Sundararaj, GJ: 2000, Exploration of the effectiveness of physical programming in robust design, *ASME Journal of Mech. Design* **22**(2): 155-163.

Chen, W, Simpson, TW, Allen, JK and Mistree,F: 1999, Satifying ranged sets of design requirements: A design metric using design capability indices, *Engineering Optimization* **31**(4): 615-639.

Choi, JK: 2002, Play locally, learn globally: The structural basis of cooperation, *Santa Fe Inst*. Available Online: www.santefe.edu.

Crow, K: 1998, *Robust Product Design Through Design of Experiments,* DRM Associates.

Fellini, R, Kokkalaros, M, Papalambros, P and Perez-Duarte, A: 2005, Platform selections under performance bounds in optimal design of product families, *Trans. of ASME* **127**: 524-535.

Gurnani, AP and Lewis, K: 2005, Robust multiattribute decision making under risk and uncertainty in engineering design, *Engineering Optimization* **37**(8): 813-830.

Hacker, KA, Eddy, J and Lewis, KE: 2002, Efficient global optimization using hybrid genetic algorithms, *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.*

Hamming, RW: 1950, Error-detecting and error-correcting codes, *Bell System Technical Journal* **29**(2): 147-160.

Hu, M, Yang, K, and Taguchi, S: 2001, Enhancing robust design with the aid of TRIZ and axiomatic design, Part 1, *TRIZ Journal*, item no. 5.

Hu, M, Yang, K, and Taguchi, S: 2001, Enhancing robust design with the aid of TRIZ and axiomatic design, Part 2, *TRIZ Journal*, item no. 5.

Jen, E: 2003, Stable or Robust? What's the Difference?, *Santa Fe Institute.*

Kicinger, R, Arciszewski, T and DeJong, K: 2005, Evolutionary computation and structural design: A survey of the state-of-the-art, *Computers and Structures* **83**: 1943-1978.

Kim, H, and Papalambros, P: 2006, Target exploration for disconnected feasible regions in enterprise-driven multilevel product design, *AIAA Journal* **44**(1): 67-77.

Luke, S and Wiegand, RP: 2002, When co-evolutionary algorithms exhibit evolutionary dynamics, *Workshop Proceedings of the 2003 Genetic and Evolutionary Comp. Conference*, pp. 236-241.

Magori, K, Orbony, B, Dieckmann, U and Meszena, G: 2003, Cooperation and competition in heterogeneous environments: Evolution of resource sharing, clonal plants, *Santa Fe Institute.*

Melvin, JW and Deo, H: 2002, Axiomatically designed robustness, *American Supplier Institute 19th Annual Taguchi Methods Symposium.*

Mitchell, M: 1999, *An Introduction to Genetic Algorithms*, MIT Press.

Newman, M, Girvan, M and Farmer, JD: 2002, Optimal design, robustness and risk aversion, *Phys. Rev. Letters* **89**(2): 028301.

Pagie, L and Mitchell, M: 2002, A comparison of evolutionary and coevolutionary search, *Int'l Journal of Computational Intelligence and Applications* **2**(1): 53-69.

Phadke, MS: 2002, Introduction to robust design (Taguchi Method), *iSixSigma Library*, Available Online: www.isixsigma.org.

Popovici, E and DeJong, K: 2002, Understanding EA (Evolutionary Algorithm) dynamics via population fitness distributions, *Proceedings of the Genetic and Evolutionary Computation Conference,* pp. 1604-1605.

Potter, MA & DeJong, K: 1994, A cooperative co-evolutionary approach to functional optimization, *Third Parallel Problem Solving From Nature*: 249-257.

Potter, MA: 1997. *The Design and Analysis of a Computational Model of Cooperative Co-Evolution,* PhD Dissertation, George Mason University, Virginia.

Sella, G and Ardell, DH: 2001, The coevolution of genes and the genetic code, *Santa Fe Institute Working Papers Library,* Available Online: www.santefe.edu.

Shelton, KA: 2004, *Development of Robust Design Concepts Using A Co-Evolutionary-Based Methodology*, PhD Proposal, George Mason University.

Six Sigma: 2001, What is Six Sigma, calculating the cost and savings of Six Sigma quality, and statistical Six Sigma definition, *iSixSigma Library*, Available Online: www.isixsigma.org.

Stadler, PF and Stephens, CR: 2002, Landscapes and effective fitness, *Comments Theor Biol* **8**: 389-431.

Wiegand, RP, Liles, WC and DeJong, KA: 2001, An empirical analysis of collaboration methods in cooperative coevolutionary algorithms, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1235-1242.

Werfel, J, Mitchell, M and Crutchfield, J: 1999, Resource sharing and coevolution in evolving cellular automata, *IEEE Trans. Evolutionary Computing* **4**(4): 388-393.

GRAMMARS IN DESIGN

An urban grammar for the Medina of Marrakesh
José Duarte, Gonçalo Ducla-Soares, Luisa Caldas and Joao Rocha

CAD Grammars: An extension to shape and graph grammars
Peter Deak, Glenn Rowe and Chris Reed

Applying evolutionary algorithm and shape grammar to generate branded produce design to meet functional requirement
Mei Choo Ang, Hau Hing Chau, Alison McKay and Alan de Pennington

A semantic validation scheme for graph-based engineering design grammars
Stephan Rudolph

AN URBAN GRAMMAR FOR THE MEDINA OF MARRAKECH

Towards a Tool for Urban Design in Islamic Contexts

JOSÉ P DUARTE GONÇALO DUCLA-SOARES AND LUISA G CALDAS
Instituto Superior Técnico, Portugal

AND

JOÃO ROCHA
Universidade de Évora, Portugal

Abstract. This paper describes research carried out to develop a parametric urban shape grammar for the Zaouiat Lakhdar quarter of the Medina of Marrakech, in Morocco. The goal is to create the basis for a system that could capture some features of the existing urban fabric and apply them in contemporary urban planning. The methodology used is described, from the initial historical analysis and fieldwork to the identification of three sub-grammars necessary to encode the complexity of the urban pre-existences: the urban grammar, the negotiation grammar, and the housing grammar. Top-down and bottom-up approaches to grammar design are analyzed and compared. The bottom-up urban grammar developed is then described, and a hand-derivation of the existing urban fabric is proposed.

1. Introduction

This paper describes research carried out to develop a parametric shape grammar able to capture, and replicate in a different context, some of the urban, architectural and morphological characteristics of the ancient fabric of the Marrakech Medina, namely its Zaouiat Lakhdar quarter. This research is part of a larger project that aims at incorporating shape grammars (Stiny and Gips 1972) with an existing generative design system based on genetic algorithms. The project's final goal is to develop a computational system able to generate novel urban and housing configurations that are more sustainable and energy efficient, while respecting certain cultural and spatial, as captured by the shape grammar. The final computational model should act

J.S. Gero (ed.), Design Computing and Cognition '06, 483–502.
© 2006 *Springer. Printed in the Netherlands.*

at two different scales: the urban scale, where the layout of an entire neighborhood is outlined; and the architectural scale, where the interior organizations of individual houses are defined. The research described in this paper is focused on the development of a shape grammar to describe the urban features of the specific quarter of the Marrakech Medina referred to above.

The reason for choosing the Marrakech Medina as the case study for this experiment was threefold. First, this particular urban fabric was attractive because of the intricate connections between the urban configurations and the patio houses, Figure 1. Second, previous work (Rocha 1995) on the Zaouiat Lakhdar, aimed at characterizing morphologically and architecturally the urban and architectural patterns of this area, suggested that a stylistically coherent corpus of designs existed and that it had enough variety and richness to fit the research objectives. Third, the population increase that occurred in Marrakech during the last decades, as in most North-African and Middle-Eastern cities, has led to an uncontrolled urban growth that produced urban environments lacking the spatial richness found in historical vernacular districts. Thus, this research intends to provide a computational framework that can assist designers in the design of urban environments that maintain traditional spatial and compositional principles while satisfying the requirements of modern life.

This research draws on previous implementation of a generative system using genetic algorithms (Caldas 2001) and on the application of shape grammars to customized mass-housing (Duarte 2001), but the work presented here takes the Marrakech Medina as its architectural precedent.

2. Historic and Cultural Context

Cities of Muslim origin, such as Marrakech (13th century), share specific culture and social values which are embedded in their everyday system of social organization, and therefore, in architecture as well. In this section, we identify and put forward a succinct contextualization of these cultural and religious values, which have to be taken in consideration in any interpretation of Islamic architecture. Social and cultural characteristics of urban planning and architecture, as well as many aspects of Islamic social behavior are related to the Islamic law, shari´ah, and certain principles found in urban environments of Islamic cities are a tribute to shari´ah. They are clearly founded in the basic source of the shari´ah, the Qur´an and the sunnah (life of the prophet), while others stem from traditional building codes related to, inheritance and endowment laws.

This set of religious public values and rules determine many of the social patterns of Islamic society and its urban and architectural spatial configurations. An utmost Islamic condition is that a strong social

relationship is underlined by the concept of brotherhood, which has frequently been mentioned in the Qur´an, and that family is the most fundamental element of Muslim society where strong family ties are expected to last.

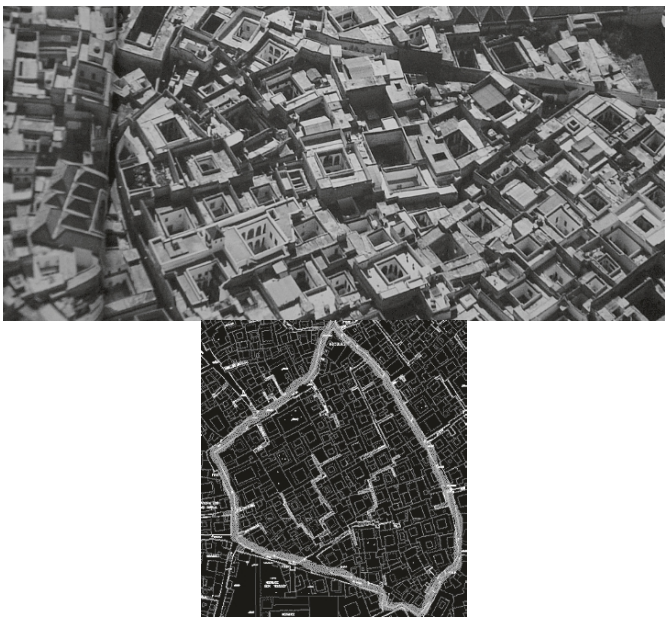


Figure 1. Aerial view (left) and plan (right) of Zaouiat Lahdar, the neighborhood selected as a case study.

This partially explains the organization of domestic architectural spaces which are close to each other and contain a multifunctional space surrounding a courtyard. It also partially explains the unsolved familiar tribal problems found in certain areas which can lead to spatial arrangements such as the closing of a Derb, the change of its direction, the destruction of a house for its division, or decisions about land division among family members and disputes of inheritance goods. Contrary to what happens in the Western world, Islamic societies do not have a precise urban code that guides the planning and design of urban environments. Islam through its shari´ah has provided principles that determine the way of life of Muslim communities and individuals, which in turn shapes the urban environment. Oleg Grabar says in his study on traditional Muslim urban environment: "it is Islam which gave resilience to the Muslim city and to its bourgeoisie, not because it was necessarily aware of all urban problems but because it had the abstract form in which all of them could be resolved (Grabar 1976)." These laws, which are constantly applied in everyday life, form a dynamic set of rules that actuate in a bottom up fashion to shape the urban tissue. This

deserves to be preserved, but at the same time, encoded in more contemporary ways of living within the Islamic culture.

3. Methodology

The methodology used to develop the computer model of the Zaouiat Lakhdar quarter, particularly the urban grammar described in this paper, encompassed three steps described below: analysis of previous work carried out to infer basic shape rules; development of an experimental computer program encoding these rules; and field work to collect additional information to complete the rules.

3.1. PREVIOUS WORK

In previous work it was hypothesized that the Marrakech Medina urban tissue, as other Islamic cities, was organized as a progression from public space to progressively more private realms, until reaching the privacy of the patio house, the predominant urban type in this part of the Medina. (Rocha 1995) The patio is the place where the outdoor activities of the family take place. The patio is also the means to provide daylight and ventilation to the house, contrarily to traditional European configurations in which the main external space is the street, and buildings are lit and ventilated primarily through street-facing facades. External facades in the Marrakech Medina are mostly closed, with very little openings. Because privacy, lighting and ventilation requirements are not significant, street width can be considerably reduced. The street thus becomes mainly a device for physically accessing the house entrance, causing it to become very narrow and often covered with constructions from the first floor on, called sabbats, thereby generating corridor-like configurations called derbs.

3.2. EXPERIMENTAL PROGRAM

Following the initial hypothesis mentioned above, several conjectural attempts were carried out to simulate what could have been the urban growth of Marrakech. A simple shape grammar composed of ten parametric rules was developed. Then, it was encoded into a computer program implemented in AutoLisp to observe the behavior of the model simulating urban growth, defined by the successive and iterative application of rules. Finally, the program was run with 50, 100, 200, 500 and 1000 iterations, Figure 2. Four problems could be observed. The first was that growth became too slow preventing the polygon representing the neighborhood to be completely filled in. The program was implemented in a way that rules were blindly applied, that is, a rule was applied and then a test was carried out to check whether it yielded a valid result. As growth evolved, it became gradually

slower to a point at which most rule applications were not valid. Consequently, it became gradually more difficult to fill in the whole polygon. The second problem was that derbs grew in all directions. Growth was constrained by restrictions imposed on the length and on the angles between rectilinear parts of derbs. Although angular values were restricted to intervals, the successive use of different values originated derbs that followed a wide range of directions, while in Zaouiat Lakhdar they tended to follow predominant directions. The third problem was that the distance between two "parallel" derbs was not controlled. In later versions of the program, a minimum distance was defined so that lots with adequate dimensions could be inserted. Results then showed that number of lots also had to be considered in determining the distance between derbs. The fourth problem was to guarantee that the limits of adjacent lots abutted. In the last version of the program, the growth of derbs was coupled with the placement of rectangles on both sides representing lots. Running the program showed that assuring that lots abutted posed a major challenge in the development of the grammar and its implementation.

3.3. FIELD WORK

To collect additional information that permitted to complete the grammar, a field trip to Marrakech took place in early 2005. Four sources of information were explored. The first consisted in surveys of the site based on laser measurements, digital photos, and hand drawings. These surveys permitted to acquire rigorous information regarding the length and width of the derbs, the height of the sabbats, and the location and size of windows and doors. The second source was documents and drawings obtained at the Agence Urbaine de Marrakech and at the Inspection Général des Monuments, such as an aerial photo of the city taken in 1950 and the plan of the Medina in digital format. The third source was interviews with local experts, which permitted to gather speculative information regarding the genesis of the neighborhood and the reasons for certain spatial configurations. Finally, the fourth source was a satellite photo of Marrakech acquired from QuickBird. The analysis of these sources of information led to the elaboration of a more accurate plan, shown in Figure 3.

4. Urban Grammar, Negotiation Grammar, and House Grammar

The view of the Medina of Marrakech suggested an organic and almost chaotic city growth. However, a close analysis unveiled a well-established order with repeated urban patterns. For example, the way lots are placed on derb corners are similar. Such patterns are not geometrically but topologically similar, meaning that they can differ in the values of parameters like the angles and dimensions of lots and derbs. Consequently, it

was possible to capture the variety of patterns into a reduced number of parametric schematas to develop a parametric shape grammar (Knight 1998).

Figure 2. Output of an experimental computer program encoding a basic urban grammar for the Marrakech Medina after 100, 200, 500, and 1000 iterations.

Figure 3. Plans of Zaouiat Lakhdar based on collected information showing the main directions of the urban fabric, and the location of derbs, sabbats, lots, house entrances, and patios (left), and which lots are accessed by which derbs (right).

At the outset, it was considered that it was necessary to deal with both the urban scale and the scale of the house. As such, the development of two independent grammars was foreseen: an urban grammar that would account for the layout of the derbs and the definition of lots, and a housing grammar that would account for the functional organization of the houses. As the

study evolved, it was realized that these two grammars could not be fully independent. In fact, the functional organization of the houses seemed to be partly responsible for the geometry of their perimeter. Moreover, the perimeter of different floors in the same house did not always match. Therefore, pre-established quadrilateral lots could not account for the generation of existing houses. This obliged to consider an interaction between the design of the houses and the urban layout and a third grammar was proposed as a result. This grammar, called "negotiation grammar," mediates between the other two grammars and regulates the permutation of spaces between adjacent lots according to the necessities of their owners. It is not certain that this "negotiation" between neighbors took place as not enough historical evidence was found. However, considering the close families ties that characterize Islamic society, to consider that it did exist seems reasonable. In fact, only a society with this tight-knit social environment could have produced such complex spatial configurations. Figure 4 illustrates the different stages reached by these grammars and the urban grammar will be described in this paper.

Figure 4. The three stages reached by the sub-grammars: urban grammar (left), negotiation grammar (center) and patio-house grammar (right).

5. Inferring and Structuring the Grammar

Given the scale of the Medina of Marrakech, the development of the grammar raised two problems, one related to the inference process, and the other to the structuring of the grammar.

In previous studies, shape grammars were developed based on a corpus of different designs within a given style. The type and scale of designs ranged from Chinese windows (Stiny 1977) to Palladian villas (Stiny and Mitchell 1978). In the current study, the goal was to develop a grammar that generated urban layouts, a design of a considerably larger scale. The Medina is composed of many neighborhoods and although these might look similar, they are quite varied in terms of morphology, street layout, urban culture, and way of living. Zaouiat Lakhdar, one of the oldest neighborhoods, has a relatively well-preserved urban fabric and its inhabitants seem to enjoy a healthy balance of safety, community life, and economic welfare. It has a

derb-based configuration with two dominant orthogonal directions. It possesses two relatively small open-spaces that are used for socializing. The unique characteristics of Zaouiat Lakhdar made it the most appealing of all the visited neighborhoods, and so it was selected as the model for the development of the urban grammar. As such, the corpus of existing designs was restricted to a single design. To overcome this limitation, the strategy used to infer the grammar was to divide the neighborhood into sectors thereby creating a corpus of several partial designs.

Structuring the grammar required a decision on whether to follow a bottom-up or a top-down approach as both seemed possible. The top-down approach provides a centralized form of controlling the design as it progresses from larger to smaller scales. In the specific case discussed here, one can imagine that a first step would be to divide the Zaouiat Lakhdar into smaller independent areas with access from the surrounding streets or from a derb. The advantage of this approach is that the relation between smaller parts is controlled from the beginning of the computation, with the shift to smaller scales only occurring when problems of larger scales are solved. This meant, for instance, that an eventual clash of derbs would be avoided because each had its pre-defined area of influence, provided that areas were defined taking the legal dimensions of lots into account.

The bottom-up approach offers a decentralized form of controlling the design, based on the independent behavior of its parts. In this approach, larger scale problems are solved by solving small scale ones. In the studied case, urban growth would be based on the individual, yet interdependent, growth of derbs coupled with the insertion of lots. Each derb would grow incrementally taking its surroundings into consideration. This approach raises some difficulties, namely, how to solve all larger scale problems, such as the clash of derbs, the alignment of lots, etc. The generation of a valid design might require an algorithm with considerable embedded intelligence. Its advantage is that it generates designs in an organic way, which might do better justice to the organic character of the urban fabric.

Not enough historical evidence was found to determine which of these approaches originated the Zaouiat Lakhdar, or whether they took place simultaneously, intertwiningly, or sequentially. Some historians (Wilbaux, 2003) do claim that the neighborhood's area was first divided into areas belonging to different families, and then each was divided into lots accessed by one derb. This would explain why many derbs are named after families (for example, derb الشلاحي, or derb الشن). Other historians claim that urban growth was spontaneous and organic; new houses would adapt to the existing situation and derbs were the empty spaces left over for access purposes. Ideally, the generation process embedded into the grammar should replicate the actual evolution of the urban fabric.

5.1. TOP-DOWN APPROACH

The top-down approach requires the decomposition of the neighborhood into smaller areas. This can be accomplished in two stages as diagrammed in Figure 5: first by recursively dividing the neighborhood into smaller areas until some condition is satisfied (steps 1-13), and then by rearranging the limits of such areas so that all can be accessed from the exterior (steps 14-20). The problem with this approach is how to divide recursively irregular forms whose division can yield forms that are topologically different from the original ones. This can be overcome by inscribing irregular forms into rectangular frames. Then these frames are divided parametrically into two or four smaller frames using one or two orthogonal lines. Then the original frame is deleted and the dividing lines are trimmed so that they do not extend beyond the limits of the neighborhood and become the limits of smaller areas. The computation proceeds until some condition is satisfied, for example, all the zones have predefined areas and dimensions that guarantee they can be accessed through derbs and divided into "legal" lots. Two types of areas will result from the computation. In peripheral areas, lots are directly accessed from the surrounding streets and host commercial functions. In inner areas, lots are accessed from derbs and are reserved for housing.

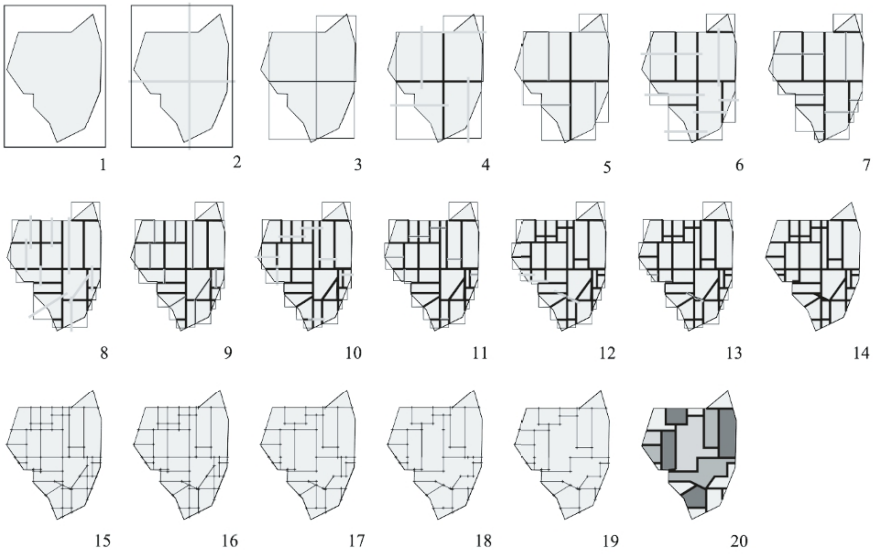


Figure 5. Top-down derivation of the Zaouiat Lakhdar zone.

5.2. BOTTOM-UP APPROACH

The bottom-up approach emphasizes the notion of growth rather than division. It requires shape rules that define both the incremental expansion of

derbs and the systematic insertion of lots, Figure 6. The basic idea is that entrance points are defined in the perimeter of the neighborhood (step 2), and then derbs grow independently from each one. Lots that have direct access from the surrounding street are defined at an early stage (step 3). Then derbs grow and fill in the empty space with lots until none is left and the whole neighborhood is packed (steps 3 through 16). The problem is that growth cannot be fully blind or independent, otherwise local voids of difficult access will be constantly created and the limits between neighboring lots will hardly ever be coincident. So, a certain level of intelligence has to be embedded in this system, both in the choice of rules to promote growth and in the way they are applied, particularly, in the assignment of values to rule parameters. There are two ways of solving this. The a priori solution requires performing local analysis to determine which rule to apply and the adequate assignment of values. The a posteriori solution implies applying a given rule with a given assignment of values and then to perform a test to check whether the inserted derb or lot clashes with existing forms; if not, it remains part of the design and the computation resumes; otherwise, it is deleted and another attempt is made using a different assignment of values or a different rule. If a bottom-up grammar is to be implemented and run automatically, it will be necessary to develop one of such higher level systems to determine whether and how growth rules can be applied.

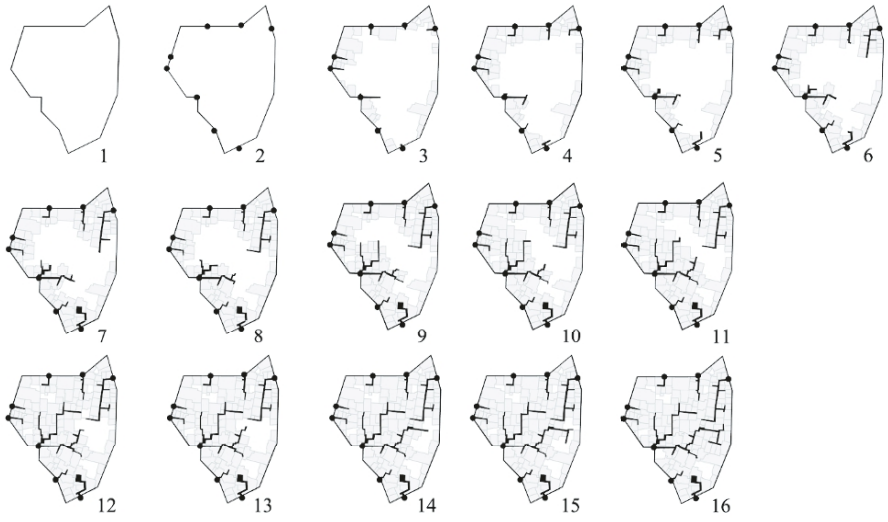


Figure 6. Bottom-up derivation of the Zaouit Lakhdar zone.

5.3. MIXED APPROACHES

The top-down and the bottom-up approaches can be combined in two different ways to develop mixed approaches. The first, Figure 7, top, uses a

top-down approach to divide the given site into different areas as described in Section 5.1, and then a bottom-up approach to pack each of the resulting areas with derbs and lots. The bottom-up stage also runs into the type of problems referred to above, but these are simplified. The second mixed approach, Figure 7, bottom, encompasses a bottom-up stage to insert derbs and their zones of influence, and a top-down stage to divide such zones into accessible lots. In this case, the problem is to avoid the clash of different zones, and so an intelligent algorithm similar to the one mentioned in Section 5.2 needs to be considered in the bottom-up stage.

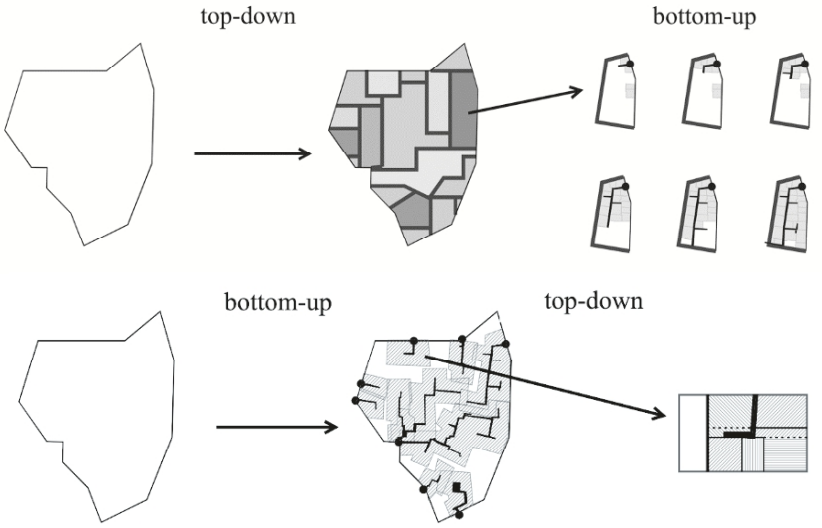


Figure 7. Top - down / bottom-up mixed approach (above); and bottom- up/top-down mixed approach (bottom).

5.4. SELECTED APPROACH

In the majority of shape grammars developed in the last thirty years, for instance the Palladian grammar (Stiny and Mitchell 1978) and the Queen Anne grammar, (Flemming 1987) designs are generated in a top-down fashion. Bottom-up approaches are more commonly used in genetic algorithms or cellular automata. Nevertheless, we believe that a bottom-up approach will reflect more honestly the organic character of the urban fabric and will eventually yield more complex design solutions. Moreover, the design of the grammar should become more challenging and interesting from a computational viewpoint. Consequently, a bottom-up grammar has been developed and is described in the next section.

6. Grammar

The proposed Marrakech grammar is a parametric shape grammar defined in the U12 algebra. The derivation of designs proceeds through six stages: (1) define the limits of the neighborhood, (2) insert entrances to derbs, (3) insert extenders and articulators forming derbs, (4) insert lots along derbs' extenders, (5) insert lots at derbs' ends, and (6) modify the layout of derbs and lots. These stages are not necessarily sequential as rules from different stages may be applied alternately. The remaining rules and further detail are available at http://www.civil.ist.utl.pt/~jduarte/dcc06/.

6.1. RULES

6.1.1. Stage 1: Define limits of the neighborhood

Stage 1 defines a polygon representing the limits of the neighborhood. This may vary in size, geometry and number of sides, but the edges should be, at least, twice as long as the minimum depth of a lot (i.e. 2 x 8 meters), and the internal angle between edges should be bigger than 60º. Two rules apply at this stage: 1.1 and 1.2, Figure 8. Rule 1.1 generates a basic triangular polygon that constitutes the initial shape and rule 1.2 introduces a vertex into an edge of the polygon so that more complex polygons can be obtained. By recursively applying rule 1.2 to the edges of an evolving polygon, a perimeter like the one that limits Zaouiat Lakhdar is obtained.

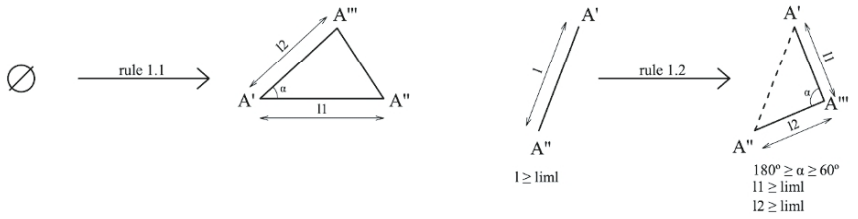


Figure 8. Rules for introducing the initial shape and defining the perimeter of the neighborhood.

6.1.2. Stage 2: Insert entrances to derbs

Rules 2.1, 2.2, and 2.3 apply at this stage, Figure 9. Each of these rules introduces an entrance point E in such a way that each edge cannot have more than two entrance-points. In rule 2.1, A'A'' is an edge of the polygon, and X' and X'' are the closest labeled points to A' and A'', respectively. They can be either other vertices of the polygon (An) or entrance-points (E). To guarantee that lots with adequate dimensions can be inserted later in the computation, the distance between two consecutive entrance points measured on the perimeter cannot be smaller than twice the length of the minimum lot depth (lim_e).

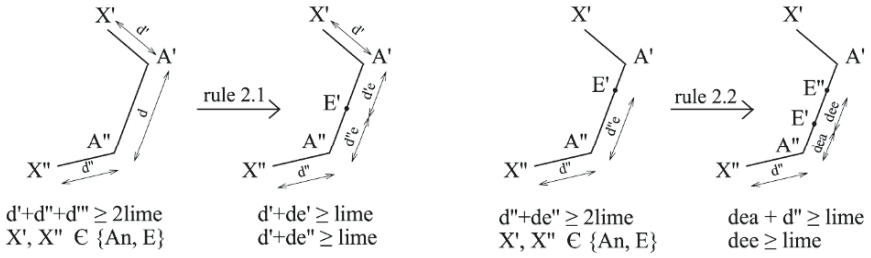


Figure 9. Two of the rules for inserting entrance points.

6.1.3. Stage 3: Insert extenders and articulators forming derbs

The third stage consists in the insertion of derbs and it encompasses rules 3.1 through 3.10. Derbs are composed of extenders that extend the derb in a pre-defined direction and are labeled with an empty triangle, and articulators that define new directions for the derb and are labeled with filled triangles. In the generation of a derb, rules that generate extenders and articulators are alternately applied, so that a extender is always followed by an articulator. Although the urban fabric inside the Zaouiat Lahdar is not orthogonal, derbs tend to follow two somewhat perpendicular directions. Therefore, the directions defined by the articulators are restricted so that the angle θ between a subsequent extender and one of the two perpendicular directions is within 20^{o}, and the angle β between sequential extenders is between 30^{o} and 150^{o}. The combination of these two restrictions defines an interval – specific to each rule application and eventually composed of non-continuous sub-intervals – within which new directions for the derb can be chosen.

Rule 3.1 inserts the initial extender of a derb, Figure 10. In this case, β defines the angle between the perimeter of the neighborhood and the direction of the extender. Rules 3.2 through 3.9 insert an articulator after a extender. They differ in the type of the inserted articulators. Rule 3.2 inserts an articulator that permits to extend the extender without changing its direction. Rules 3.3, Figure 11, and 3.4 insert elbow-like articulators, and rules 3.5 and 3.6 t-like articulators. Rule 3.7 inserts an arrow-like articulator, rule 3.8 a y-like articulator, and rule 3.9 a cross-like articulator. Finally, rule 3.10 connects a extender to an articulator marked with a filled triangular label. The parameters in this rule are the length (l) and the ending width (w) of the extender.

6.1.4. Stage 4: Insert lots along derbs' extenders

In Stage 4, lots are inserted along derbs. In most cases, these are almost rectangular and, in the proposed shape grammar, this quadrilateral shape is captured by a general topological schema with specific constraints. The value of internal angles and the dimensions of their sides may vary between

specified intervals. Furthermore, the proportion of the lot is confined to the interval between 1:1 and 2:3 meaning that its geometry may vary from a square to a rectangle in which the length cannot exceed 1.5 times the width. Not all lots are quadrilaterals as some may have more than four sides. However, it is possible to identify a main quadrilateral shape in such lots, which is then used for matching the schema on the left-hand side of rules. As lots follow the general topological schema just described, the internal shape parameters are not shown in the rules for inserting lots for simplification purposes.

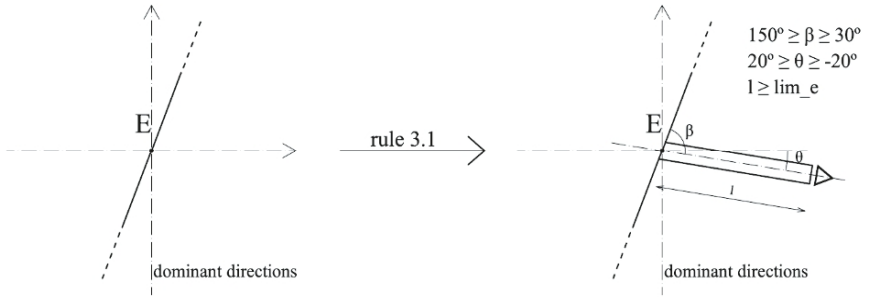


Figure 10. Rule for inserting the initial extender of a derb.

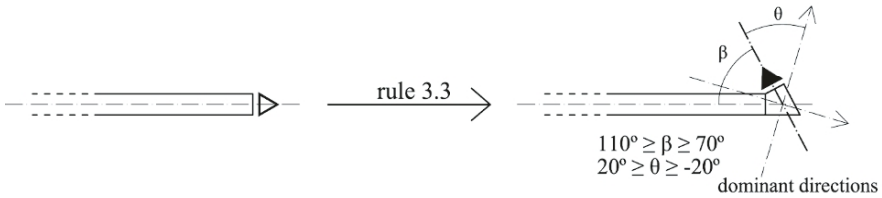


Figure 11. Example of rule for inserting an articulator.

Rules 4.1 through 4.5 define the insertion of lots along extenders. Rules 4.6 through 4.13 define different forms of inserting lots on the outer side of elbow-like articulators, and rules 4.14 through 4.16 do the equivalent on the inner side. Finally, rules 4.17 through 4.21, deal with the situations created by other articulators. Rule 4.1, Figure 12, top, is the seed-rule at this stage as it is responsible for the insertion of the first lot of the derb and it takes into account the perimeter of the neighborhood. The rule has five parameters: width w, distances d1 and d2, and angles α1 and α2. Rule 4.2, Figure 12, bottom, inserts a new lot along an extender that has not been completely filled in yet. An extender is identified as empty or full through labels E or F, respectively. Rule 4.2 only applies to situations where the extender has the label E and it keeps it unchanged. Rule 4.3 does the same as rule 4.2 except that it fills the extender, thereby changing the label to F. Rules 4.4

and 4.5 are applied when the extender' available length is smaller than the lot's minimum width. Rule 4.4 changes the label to F without adding a lot. Once an elbow-like articulator has been added to the extender, rule 4.5 introduces a lot that stretches beyond the limit of the articulator.

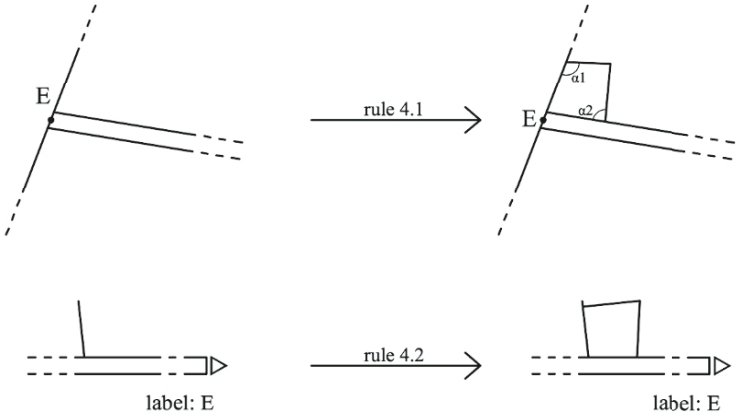


Figure 12. Example of rules for inserting lots along extenders.

Rules 4.6 through 4.13, deal with all the possible situations that can be created by the introduction of elbow-like articulators. In the application of parametric rule 4.6, Figure 13, three parameters, e, $\gamma 1$, and $\gamma 2$, need to be assigned values; e is the distance of the limit of the lot to the limit of the articulator. It is measured on an auxiliary axis, whose origin is O, with positive values on the left. Depending on the value of e, angles $\gamma 1$ and $\gamma 2$ can take one of two values to align the limit of the lot with the incoming extender or the out-going extender. If $e \leq 0$, then the angle $\gamma 1$ is not defined and $\gamma 2$ can have one of two values: 90° or 180° - β, with β defining the angle between the two extenders. If $e > 0$, then $\gamma 1$'s value can be either 90° or 180° - β. If $\gamma 1 = 90°$ (i.e. the limit is perpendicular to the in-coming extender), then $\gamma 2$'s value can be either 90° or β° (i.e. the lots' limit is either parallel to the in-coming extender or perpendicular to the out-going I-extender). If $\gamma 1 = 180° - \beta$ (i.e. the lots' limit is perpendicular to the out-going extender, then $\gamma 2$'s value can be either 90° or 180° - β (i.e. the lots' limit is either perpendicular to the out-going extender or parallel to the in-coming one). The closed polygonal shapes of the lots are not fully represented in the rule. As the rule specifies the relation between the lots and the extenders to which they are connected (in this case, an in-coming extender, an elbow-like articulator and an out-going extender) how the polygon is closed is not important. The remaining rules follow a similar scheme. Note that in all the rules, angle $\gamma 2$ is such that the limit of the lot is parallel to the in-coming extender or perpendicular to the out-going extender.

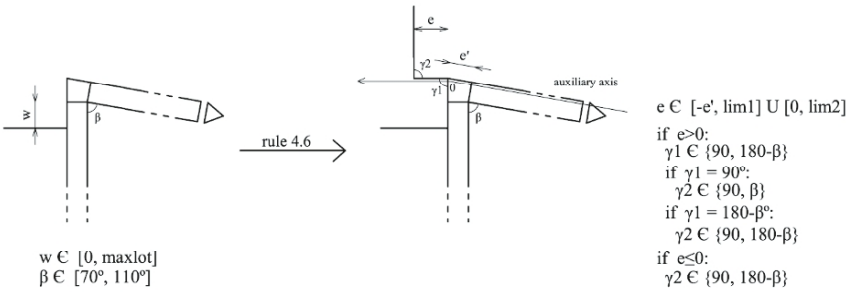


Figure 13. Example of rule for inserting lots on elbow-like articulators.

Rules 4.14, 4.15, and 4.16, are used to fill inner (concave) corners (corners defined by the intersection of one in-coming and one out-going extender) with lots, Figure 14. These corners need not be exclusively derived from the insertion of elbow-like articulators; any corner with an angle smaller or equal to 90° can be tackled with these rules regardless of the specific articulator involved. The problem is addressed in the following way. Rule 4.14, places a label P on the vertex of the last lot that is more distant from the corner between the in-coming and out-going extenders. If the distance d between point P and the out-going extender is larger than the minimum width of a lot, then rule 4.15 is applied to transform the lot into a corner lot. If it is smaller, then rule 4.16 transfers the label P to the last but one lot, and deletes the last lot. This rule is applied recursively until Rule 4.15 can then be applied.

Rules 4.17 and 4.18, Figure 15, insert a lot in an outer corner (corner defined by the intersection of two out-going extenders), which may be yielded by rules 3.5, 3.6 and 3.9. Four variables are involved in these parametric rules: w1 and w2 are the front dimensions of the lot and they can take any value in the interval defined by minlot and maxlot (respectively 8 m and 18 m in this case); and $\alpha 1$ and $\alpha 2$ are the angles between the limits of the lot and the extenders, which can vary from 70° to 110° each.

Rule 4.19 defines the insertion of lots in the specific case where the in-coming direction coincides with one of the two out-going ones. In this case, a continuity of extenders is observed and may be generated by rules 3.5, 3.6 and 3.10. For this rule to be applied, the distance e between the right most limit of the last lot and the articulator has to be inferior to a given lim. Then, the values of the parameters' on the right-hand side of the rule must be satisfied: $\alpha 1$ and $\alpha 2$ can vary from 70° to 110°, w1 must be positive, and w2 and d (width and depth) must be between minlot and maxlot. Finally, rules 4.20 and 4.21 handle the insertion of lots on the sector defined by the intersection of the out-going extenders, whenever rule 3.7 or rule 3.8 have been previously applied.

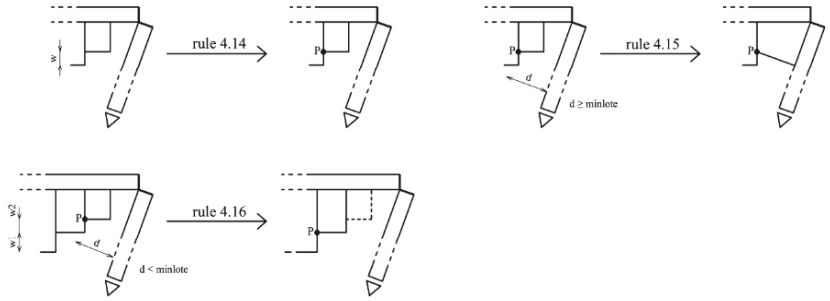


Figure 14. Rules to introduce lots in the inner corner formed by extenders.

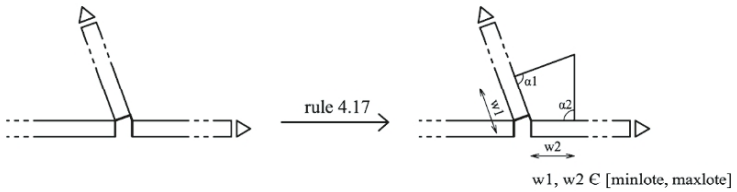


Figure 15. Example of rule to insert lots in the outer corner formed by extenders.

6.1.5. Stage 5: Insert lots at derbs' ends

Stage five deals with the insertion of lots in the ending extender of a derb. There are six possible layout configurations that can be found on the left-hand side of the rules in this stage. For each of these configurations, there are several ways in which lots can be placed at the end of the extender and these are encoded into the right-hand side of the rules. Rule 5.1 is shown in Figure 16. This rule inserts three lots at the end of the derb. For the rule to be applied, the positions of the last lots on both sides of the extender must be such that the distances between their limits and the end of the extender, respectively d1 and d2, are smaller then the minimum dimension permitted for a lot, minlot, which means that no further lots could have been placed using rule 4.2. The remaining rules (rules 5.2 through 5.19) work in a similar fashion.

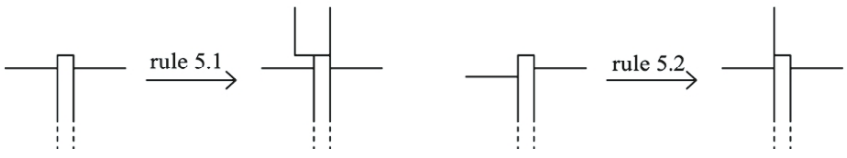


Figure 16. Example of rules for inserting lots in the ending extender of a derb.

6.1.6. Stage 6: Modify the layout of derb's and lots

Finally, stage six encompasses rules that modify the existing lots to create smaller or larger lots, to access locked lots, or to reach locked empty spaces

to resume growth. The modifications introduced by these rules in the lots respect the general topological scheme described in Section 7.2. As such, the specific constraints on the shapes of modified lots are omitted in the rules. Rule 6.1 divides a larger lot into two smaller ones. Rules 6.2 and 6.3 expand a lot at the expense of an ending extender (literally a dead-end alley.) Rule 6.4 inserts an extender perpendicular to an existing extender at the expense of a lot to provide access to a locked lot or to an empty locked area. Rule 6.5 through 6.7 are similar, except that in rule 6.6 the new extender is aligned with the existing one, and in rules 6.6 and 6.7 a change of direction requires the introduction of an articulator. Rule 6.8 is similar except that its purpose is exclusively to provide access to locked lots and no further growth of the derb is foreseen. Rule 6.9 also provides access to a locked lot, but by modifying its topology at the expense on adjacent lot, instead of inserting an extender. Rule 6.10 provides access to a lot that is connected to a derb solely through a vertex. In this case the topologies of two lots are modified so that an entrance area is generated off the lot that is already accessible through the derb. Finally rule 6.11 connects two derbs by means of a diagonal direction.

6.2. PARTIAL DERIVATION OF THE EXISTING ZAOUIAT LAKHDAR

The grammar just described is non-deterministic and open-ended. In general, more than one rule can be applied at a given step in the derivation. Furthermore, a single rule can generate different solutions depending on the values assigned to parameters. This means that from the same perimeter different applications of the grammar rules will likely yield different solutions. Consequently, the application of the grammar generates unpredictable results. Figure 17 shows the last steps in the generation of the upper part of the existing Zaouiat Lakhdar neighborhood using stage 4 and stage 6 rules. Step 1 depicts the state of the design at the end of stage 5. Step 2 results from the application of rules 6.1, 6.2 and 6.3. Step 3 results from the application of these and rule 6.4. In step 4, additional lots are added using stage 4 rules. In steps 5 and 6, rules 6.5 through 6.11 are applied to complete the layout.

7. Discussion and Conclusions

The research described in this paper constitutes one step towards the development of a computational model of the Zaouiat Lakhdar neighborhood in Marrakech. The ultimate goal is to use this model in the planning and design of new neighborhoods that have similar spatial features and yet are improved from the environmental viewpoint. The model uses shape grammars to encode the underlying syntactic rules and genetic algorithms to “optimize” solutions. It encompasses three grammars: a grammar to generate the urban fabric, a grammar to generate the houses and

a grammar to trade spaces among adjacent houses. This paper describes the first of these grammars. In the next sections the limitations of the current grammar are discussed and future work is outlined.

Figure 17. Different steps within stage 6.

The current grammar is bi-dimensional, but traditional urban environments in Islamic cities present three-dimensional complexity. In fact, the morphology of an Islamic city such as Marrakech cannot be described as the simple extrusion of bi-dimensional forms defined in plan. Its variety is just as rich in section as it is in plan. Consider, for instance, the Sabbats that cover the derbs. In addition to constitute a rich architectural feature of great formal plasticity, they exist for several reasons of which some are to provide structural stability to nearby houses, to create shade for environmental comfort, and to extend housing spaces to fulfill family needs. Another feature with similar impacts is the trading of spaces among adjacent houses which causes the perimeters of different floor plans not to coincide. Features like these cannot be fully described in two dimensions, but have important impacts on functional organization and environmental performance. Therefore, they are important for the type of "optimization" targeted with the model, and so future work will be concerned with the extension of the current grammar to include three dimensions.

One of the issues raised by the adoption of a parametric urban grammar concerns the criteria for choosing values for rule parameters. One interesting possibility is the drive to improve some performance indicators, thereby guiding the solution towards certain desirable characteristics. The shape grammar presented in this paper will be coupled with a genetic algorithm

(GA) to form a generative design system that performs guided search for improved urban patterns, a term we prefer to that of optimization.

This guided search may act at the urban scale, where potential fitness functions for the GA may be related to issues of density, ratio of public vs. private space, maximum length of derbs, and so on. Guided search may also act at the level of the private patio houses, by improving the environmental performance of the houses and providing modern living standards in terms of day-lighting, ventilation, thermal performance, and other environmental design parameters. Given a certain lot, determined after the application of the urban grammar, many design choices will have a deep influence on the future performance of the house, such as patio configuration, spatial layout, loggia design, type of façade and roof, openings design and layout, construction materials, colors and external finishes, among others.

Although the current study is based on the study of the Medina of Marrakech, the ultimate goal is that by introducing variations in the grammar rules, the model might be applied to new city districts not only in Marrakech, but also in other cities throughout the Arab world. Because of the demographic boom, the shortage of qualified personnel, and the scarcity of funds, we would argue that this tool is particularly appropriate for use in the design and planning of cities in this region.

Acknowledgements

This research was carried out within the framework of the project POCTI/AUR/42147/2001, with financial support of FCT, Portugal.

References

Caldas, LG: 2001, *An Evolution-Based Generative Design System: Using Adaptation to Shape Architectural Form*, PhD Dissertation, Massachusetts Institute of Technology.

Duarte, JP: 2001, *Customizing Mass Housing: A Discursive Grammar for Siza´s Malagueira houses*, PhD Dissertation, Massachusetts Institute of Technology.

Flemming U: 1987, More than the sum of parts: The grammar of Queen Anne houses, *Environment and Planning B: Planning and Design* **14**: 323-350.

Grabar, O: 1976, Cities and citizens: The growth and culture of urban Islam, *in* B Lewis (ed), *Islam and the Arab World*, Thames and Hudson, p. 100.

Knight TW: 1998, Designing a shape grammar: Problems of predictability, *in* JS Gero and F Sudweeks (eds), *Artificial Intelligence in Design,* Kluwer, Dordrecht, pp. 499-516.

Rocha, J: 1995, Marrakech: An evolutionary model, *Abstract*, New York, Columbia University Press.

Stiny, G and Gips, J: 1972, Shape grammars and the generative specification of painting and sculpture, *in* CV Freiman (ed), *Information Processing 71*, North-Holland, Amsterdam, pp. 1460-1465.

Stiny G and Mitchell, WJ: 1978, The Palladian grammar, *Environment and Planning B* **5**: 5-18.

Wilbaux, Quentin, 2001: *La Medina de Marrakech. Formation des Spaces Urbains d´une Ancienne Capitale du Maroc*, L´Harmattan, Paris.

CAD GRAMMARS

Combining CAD and Automated Spatial Design

PETER DEAK, GLENN ROWE AND CHRIS REED
University of Dundee, UK

Abstract. Shape grammars are types of non-linear formal grammars that have been used in a range of design domains such as architecture, industrial product design and PCB design. Graph grammars contain production rules with similar generational properties, but operating on graphs. This paper introduces CAD grammars, which combine qualities from shape and graph grammars, and presents new extensions to the theories that enhance their application in design and manufacturing. Details about the integration of CAD grammars into automated spatial design systems and standard CAD software are described. The benefits of this approach with regards to traditional shape grammar systems are explored.

1. Introduction

The aim of the Spadesys project is to investigate how spatial design can be automated in a generalized way, by connecting similar concepts across the various design domains and decoupling them from the intelligent design process. The primary focus is on engineering design domains, where there is a large number of domain specific constraints and requirements, as well as problem specific constraints and requirements for each design being produced.

Shape grammars have proved to be applicable in a range of different design domains from camera to building design, which sets them as an appropriate technique to further the goals of generalized design. They employ a generative approach to creating a design using match and replace operations described by a grammar rule set for a domain. There are, however, a number of issues or limitations associated with shape grammars:

- Engineering domains will have a large set of inherent domain requirements, and each specific design to be generated will have a large set of problem specific requirements and constraints related to

J.S. Gero (ed.), Design Computing and Cognition '06, 503–520.
© 2006 *Springer. Printed in the Netherlands.*

that instance. Creating a grammar rule set that contains the maximal amount of domain knowledge, while remaining flexible and adaptable enough to fulfil the greatest number of designs can result in a large or complex grammar rule set.

- Communicating grammar effectively is difficult; justification for individual grammar rules can be difficult to provide, as they may not have a direct significance on a design, instead playing a linking role where they prepare parts of the design for further grammar rules to work on. This can make maintenance, and understanding of the grammar by anyone who was not involved with its creation difficult.
- In order to use shape grammars in an automatic design generation scenario in most engineering domains, the grammar has to be very detailed and complete, and prohibit the introduction of flaws into the design.
- It is difficult to verify a grammar. A recursive rule set can define an infinite space of possible solutions, and can therefore contain designs that may be flawed in ways that were not anticipated by the grammar designer.
- Current shape grammar implementations do not make it possible to express connectivity; if two line segments in a design share a common endpoint, it is not possible to show whether they are segments of a logically continuous line, or two unrelated lines which happen to be coincident.
- It is Difficult to create a 'designerly' grammar, where the order and application of rules proceeds and a way that makes sense to the user.

2. Graph Grammars

Graph grammars (Plump 1999) consist of production rules to create valid configurations of graphs for a specific domain. They have been successfully employed in designing functional languages (Barendsen 1999) and generating picturesque designs (Drewes 2000). Graph grammar rules contain the match and replace operations for nodes and edges in a network.

There is generally no spatial layout information associated with the nodes and edges; the only relevant data is the types of nodes and edges, and the information about the connections between them. It is therefore difficult to model spatial and graphical designs with graph grammars alone. A desirable feature with graph grammars is that the application of grammar rules keep the design connected as the network is increased.

3. Shapes and Graphs

In typical CAD applications, some of the primitives used to model designs are vertices (points in 3D space), edges (lines connecting points), and faces (enclosed polygons made by edges). This has proven to be an effective way

of representing many types of spatial data, as it allows for a range of editing and analytical operations to be applied to a model. Vertices represent a sense of connectivity between lines. This makes it helpful to display and edit designs and express relationships between lines. Traditional shape grammar systems are not able to deal with CAD primitives directly. Using a design from a CAD application in a shape grammar system would require conversion of the designs representation to be compatible with the components of the specific system. It would be desirable if the representation does not have to be altered from the one used in CAD software.

There is a clear correlation between these CAD elements and graphs. A design represented using CAD elements can be seen as a graph, with the vertices being the nodes of the graph and lines being the arcs or edges. A CAD design is more complex however, and contains more information, as not only the presence of nodes and arcs, but also their positions and lengths are relevant. Graph grammars have been used in a similar way to shape grammars to design graphs, and an advantage of graph grammars is that there is a sense of connectivity between the elements.

In the Spadesys system, one of the core ideas is to combine shape grammars with graph grammars, inheriting the beneficial features of both concepts. Additionally, in Spadesys there are a number of extensions and new possibilities which are not found in any other shape or graph grammar system. “CAD grammars” are thus an amalgam of the two systems, and inherit benefits from both. In order to address remaining limitations, a number of extensions are proposed, and their implementation in Spadesys is discussed.

4. CAD Grammar Fundamentals

Rules in CAD grammars are comprised of two parts, the match shape, which is a specification of the shape to be matched, and the replace shape, which is the shape to replace the specified match shape. The design shape is the specification of the current design that is being generated. The matching algorithm looks to find occurrences of the match shape within the design shape, and replace those configurations with the replace shape.

The basic elements for shapes in a CAD grammar system are *points* and *lines*. *Points* are objects which have the numerical parameters x, y (and z in a 3D implementation). Lines are represented by references to two points; *p0* and *p1*. It is important to consider points and lines as objects; as there may be multiple points with the same parameters, but are distinct entities.

Connectivity among two lines can be represented by the two lines sharing a common point instance. In CAD grammars it is important to be able to make this distinction in the design shape and the match/replace shape. The usefulness of this feature can be seen in instances where two lines happen to

appear to share an endpoint, but they are not intended to be logically continuous with regards to the grammar matching algorithm.

The following is an example of the connectivity features of CAD grammars:

Point1: X=2, Y=3
Point2: X=4, Y=4
Point3: X=5, Y=6
Point4: X=2, Y=3
Point5: X=4, Y=4
Point6: X=4, Y=4
Point7: X=5, Y=6

Continuous, connected lines:
LineA: Point1, Point2
LineB: Point2, Point3

Non-connected lines:
LineC: Point4, Point5
LineD: Point6, Point7

In Figure 1(a), the two line segments are connected, which can be seen by the use of only three point instances, with Point2 being shared by both line segments. Figure 1(b) shows spatially identical, non-connected lines, with each line having unrelated point instances.

Visually, both LineA⇨LineB and LineC⇨LineD appear similar:

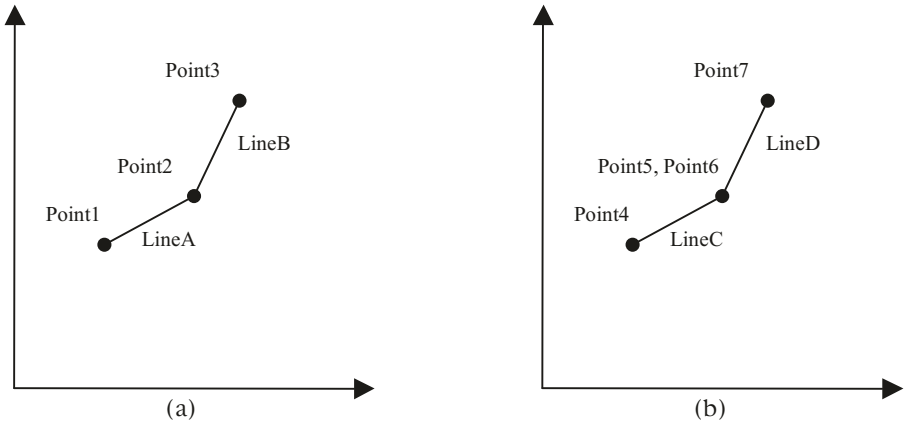


Figure 1. Line Connectedness.

Similarly, intersecting lines do not logically subdivide into four line segments, unless this is specified as the intended operation of the grammar

rule, by setting properties of lines in the replace shape to join-and-divide at intersections. The reason for this is that there are many cases when the result of applying certain grammars results in lines intersecting, but it is not the intention of the grammars to have the intersection produce corners, which are matched by other grammar rules. This can prevent accidental, unintended matches in further operations on a shape. For example, the match shape in Figure 2(a) would successfully match the design shape in Figure 2(b), but not that in Figure 2(c).

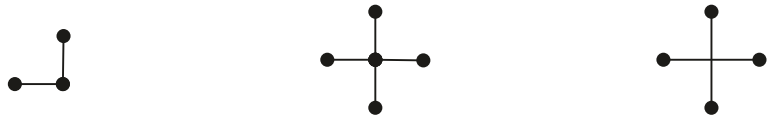


Figure 2. Matching connected lines.

5. Extension 1: Length and Angle Constraints

Parametric shape grammars (Stiny 1980) allow specification of variable parameters in shape grammars. In Spadesys's CAD grammar, the definition of parameters and their usage is enhanced. Similar work has been done in Liew (2004).

Every line in a match shape can have a length constraint. This length constraint is evaluated with the line that is to be matched in the current design when running the matching algorithm.

With regards to many engineering design domains, there may be a need to specify exact line sizes in the match shape, which will result in lines only of that exact length being matched. In CAD grammars, if the length constraint for a line is an exact value, then that line will only match lines of that value. This allows match shapes to be drawn inexactly when actual values are known for the line lengths. Similarly, the length constraint may be specified as a range such as 4-10, in which case all lines of length between 4 and 10 will be matched. Logical operators can be used within the length constraint to allow further control on matching; for example we want to match lines of length 7 or 15, we can set its length in the match shape to 7 | 15. Similar constraints can also be applied to angles between lines, to provide similar flexibility with regards to appropriate angles too.

When the length constraint is set to *proportional*, the behaviour is similar to most traditional shape grammar systems, where any line length will match, provided that all the lines which were matched have the same proportions as the lines in the match shape, making the scale of the match shape irrelevant. When the length constraint is set to *length*, then the exact length of the line is used, as it is shown graphically in the match shape. This

is different from exactly specified lengths, as they may be a completely different size from the physical length of the line in the shape.

Due to the complete scripting system embedded within Spadesys, complex mathematical operations can be also used in the length constraint.

6. Extension 2: Modification

Shape grammars (as well as all formal grammars) operate using match and replace operations only. When the aim of a grammar rule is to modify a feature, it is achieved by having a similar match and replace shape, which vary in terms of the intended modification. In standard shape grammars, this approach is fine, since there is no difference between actually modifying the matched shape's elements in the current design, or simply removing it and inserting a new shape as desired. However in CAD grammars there can be meta-information associated with the lines and points in a design, which in many cases would need to be retained.

The most important part of the meta-information of a line is its connectedness; i.e. which other lines it is connected to. It is necessary to be able to state in a grammar rule whether the elements in the current shape should be replaced by new instances of the elements in the replace shape, or whether they should be modified as stated by the elements in the replace shape. The effect of this idea in practice is that grammar rules can not only match and replace, but they can also *modify*. This means that there can be two grammar rules that look identical with regards to the lines and points, but create a completely different result when applied. This is unlike the effect of modification that can be achieved using only match and replace, as seen in the following examples.

The grammar rule in Figure 3 is designed to stretch the match shape regardless of context.

Figure 3. 'Stretch' shape grammar rule.

When applied traditionally to the following example, unintended results are produced, so that the design shape in Figure 4(a) changes to the shape in Figure 4(b), rather than what was intended: Figure 4(c).

To get the intended result with a traditional shape grammar approach, there would need to be a larger, more complex grammar that takes into account all possible contexts of the original match shape, and modifies the effected portions of the design shape separately. In Spadesys, the above grammar rule from Figure 3 would be represented as the rule in Figure 5.

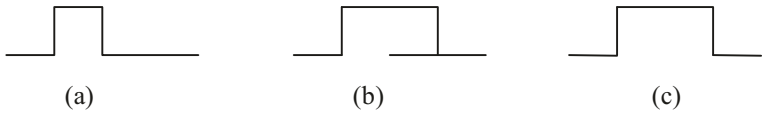


Figure 4. Tradition application of rule.

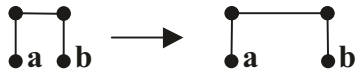


Figure 5. Connectedness in matching.

This modification ability is currently implemented using a tagging system. The points in the match and replace shape can be tagged with labels (strings) to signify their correspondence. In Figure 5, the 'a' and 'b' labels associated with the points represent their tags. If a point in the replace shape has the same tag as a point in the match shape, then the matched point in the design shape will be modified to spatially match the replace point, as opposed to removing it and replacing it with a new point. This ensures that the connectivity of the point in the design shape is maintained after the replace operation, and gives the effect of modifying the shape, as opposed to deleting and inserting portions.

7. Extension 3: Line Types

Initially, non-terminals in shape grammar systems have been represented by a combination of terminals that is unlikely to be found elsewhere in the shape (only in the parts where it is intended to be a non-terminal). This requires complicating the design, and is not safe or efficient. Colours (Knight 1989) or weights (Stiny 1992) can be added to grammar rules to improve this method, but Spadesys introduces polymorphic and hierarchical 'Line types' as a parameter for lines in shapes. Types are hierarchically structured entities in the same sense as classes and subclasses are in programming languages. The base type is *Line*, and all other line types derive from it. Due to the polymorphic nature of types, if a line in a match shape is of type *Line*, then it will match any type of line in the current design (provided the length constraint is also met).

Generative design often takes place in phases (Stiny 1978), by gradually lowering the level of the solution from a high-level/abstract design to a low-level/complete design, until it satisfactorily represents the requirements. For example in architecture, the solution can initially start off as a grid of squares covering the approximate layout of the intended design. Applying an

initial grammar set in the first phase will add some temporary walls to outline a basic conceptual layout. The next phase can add more detail on the shape of the walls, and position them adequately. Further phases may add additional details such as doors or windows, and so on. By annotating the lines in grammars with their types, we can show clearly which grammars should be applied at the first phase (by setting the match shapes lines to type *grid*) and what phase it will prepare its results for (by setting the replacement shapes lines to type *basicwall*). This opens up more flexible approaches with regards to the progression of the shape generation. One half of the building can be generated right up to the windows and doors phase, and once satisfactory, the other half may be worked on without interference. This region based workflow may be more appropriate in some cases than a phase based one.

The polymorphic nature of types allows control over the generality of grammar rules: from being applicable to any relevant part of a design (when the line type is set to the base type 'Line') to domain or problem specific locations. Grammar designers can incorporate this into the abstraction of the operations; create domain independent rules such as a rule that extends the dimensions of a rectangle (Which can apply to lines of all types), to domain specific rules such as a rule that adds an alcove to lines of type 'WoodenWall' and its derivatives.

Grammar interference is also removed, and the grammars from different phases do not have to be handled separately. A grammar rule will only be applied where it is intended to be applied, on the types of lines it is intended to be applied.

A grammar rule becomes self documenting to an extent, as the line types describe when and where it is applied, and more accurately shows what the designer is trying to achieve with the grammar rule.

8. Partial Grammars

Spadesys attempts to drive the use of partial grammars as a way to tweak and modify designs in a clear and simple way. When the aim is to modify existing designs with new features, it may be inefficient to determine their grammar rule set and modify it in a suitable way so that when the design is re-generated it contains the intended changes. It may be simpler having a partial grammar containing only the rules for the new features, and applying that to modify the design. A partial grammar is a reduced set of grammar rules with the intent to modify existing designs, rather than generate a complete design from nothing.

For example, given the existing design in Figure 6(a), the aim is to round off the edges to produce Figure 6(b).

Figure 6. An example of modification.

The complete grammar would either have to contain all the rules to produce the source shape with the addition of rules to perform the modification, or the rules would have to be modified so that the intended design is produced directly. Either way requires the original grammar, which may not exist and can be difficult to derive. In Spadesys, the grammar rule similar to the one in Figure 7 can be directly applied to any design shape.

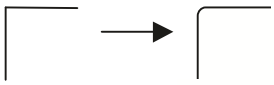


Figure 7. A rounding rule.

The application of the rule in Figure 7 on the design shape in Figure 6(a) demonstrates another useful feature of CAD grammars that derives from the extended connectivity features. Without the connectivity information in the design shape, automatic application would require the original shape to contain additional shape labels regarding which corners should be treated as ones to be rounded; otherwise unexpected results can be produced, such as the one in Figure 8.

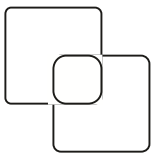


Figure 8. Incorrect rounding.

But with the CAD grammar elements, the initial design shape would be represented as shown in Figure 9. Using lines and points as components in the design means that additional labeling is not required, as the intention of corners and intersections is implicit. Liew (2004) presents an alternative method for controlling the application of grammar rules.

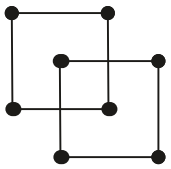


Figure 9. CAD grammar representation.

Partial grammars may also be used as the basis for generating the design. In an architectural example, a grammar for a house may be designed in a way, that it is incomplete, and cannot generate a design on its own. However, when provided with a building outline as the current design, it can generate the remainder of the house.

The modification features of CAD grammars as described above in extension 2 are very assistive to the idea of using partial grammars. The modifications can be represented in a compact, context free way by being able to preserve connections between lines and therefore modify the surrounding context suitably. The length constraints feature as described in extension 1 is also a valuable feature for such situations, because a single grammar rule becomes more flexible and can apply to more varying configurations.

9. Implementation

9.1. CAD GRAMMAR SYSTEM: MATCHING ALGORITHM IMPLEMENTATION

The matching algorithm is used to determine if, and where the match shape in a grammar rule can be found in the current design. It is a recursive algorithm that is carried out on each line of the current design. This is the pseudo code of the matchLine algorithm:

matchLine(Line source, Line match)

1. If we have already passed this match line, return true
2. If the number and angle of every branch of match is not the same as the number and angle of every branch of source, return false
3. If match's length constraint does not prove valid with source, return false
4. If source's type is not of match's type or a subtype of that, return false
5. For each corresponding branch of match and source
    a. matchLine(sourceBranch, matchBranch)

This algorithm attempts to traverse the match shape and the current segment of the design shape in a mirrored way by passing and attempting to match each line and their branches recursively. Visited lines are ignored, which breaks up the traversal and prevents infinite loops in circular designs. This also results in the shapes (which have the connectivity structure of a graph) to be parsed as a tree.

9.2. CAD GRAMMAR SYSTEM: REPLACEMENT ALGORITHM IMPLEMENTATION

The replacement phase occurs when the match shape has been found at a location in the design shape. An important factor to consider is that the replacement shape must have the same scale and alignment as the match shape has in the design shape as it was located. Therefore, the transformation matrix between the match shape and its corresponding shape in the current design has to be determined. The transformation matrix encapsulates the scale, rotation and translation that is required to convert the replace shapes alignment to the correct configuration. The first line from both the match and the design shape is used as the point of reference (the first line can be specified, but by default it is the line that was placed first), and the transformation matrix is determined from these lines only. The scale and rotation can be obtained directly from the two arbitrary lines. The translation can then be obtained by applying the scale and rotation to the line from the match shape, and finding the offset between the corresponding endpoints of the match and source line.

This matrix is applied to the replace shape before its insertion into the design shape. During insertion, if any points have a matching tag in the match shape, then the point in the source is modified to correspond to the point in the replace shape, rather than replaced.

10. CAD Grammar Applications

10.1. AUTOMATED DESIGN GENERATION – SPADESYS

At the core of the Spadesys project is the CAD Grammar, which provides an implementation of the theories described above. There is no tie to any of the phases of the design process (Reffat 2002); a base design can be generated automatically and exported for further manual refinement; an existing base design may be imported for detailing; or the entire process may take place within the software. The application and use is all down to the grammar sets in play, which may be domain independent (simple CAD-like modelling operations) or domain dependent (grammar to generate buildings/phones/circuit boards etc.).

As demonstrated in Figure 10, the first step is defining the problem that is to be solved. This involves producing problem code, which details the

constraints and requirements of the design that is to be generated. The code is converted into a native representation that can be loaded into the intelligent designer. An existing design may also be imported or created by the user based on the grammar set to be used. Partial grammars may require an existing design of some form to operate on. The grammar rule sets that are to be used must also be specified. These can be made up of domain independent basic construction grammars, but will most likely contain domain dependant sets that contain grammar rules specifically for the generation of the intended artefact.

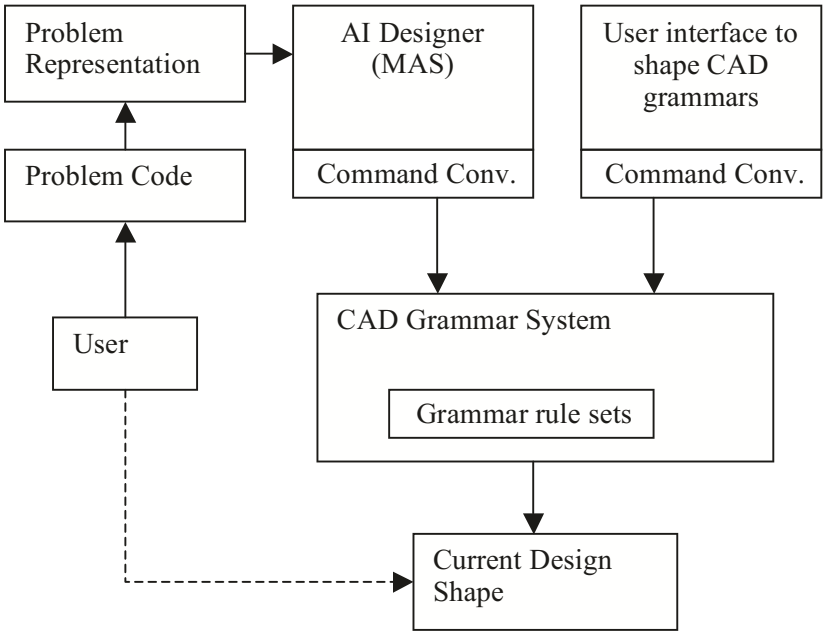


Figure 10. Architecture of Spadesys.

The intelligent designer then uses the data from the problem code to make decisions regarding the design's generation. It is currently implemented as a multi-agent system, but can be replaced in the future with a different kind of *reasoner*.

The command conversion layer for the intelligent designer translates decisions made by the intelligent designer into commands for the CAD grammar system, such as 'apply grammar rule *x* at location *y*, with these parameters'. The command conversion layer for the user interface does a similar job by taking the actions of the user and converting that into the intended commands. At its core, it is the same shape grammar system

performing the same process, without any knowledge as to whether it is a human user or the intelligent designer making the decisions.

The Spadesys application provides a feasible motive for the design of CAD grammar rules.

Figure 11 shows the entire interface with the rounding example from Section 8. The main user interface to the application is user configurable, with all windows being dockable to any portion of the screen, in the same way as many Integrated Development Environments (IDEs). The top half of the central viewport has a grammar rule set opened with the rule to round off edges. The bottom half contains the design shape which is to be modified. All shapes can be edited by the user using the provided CAD-like toolset.

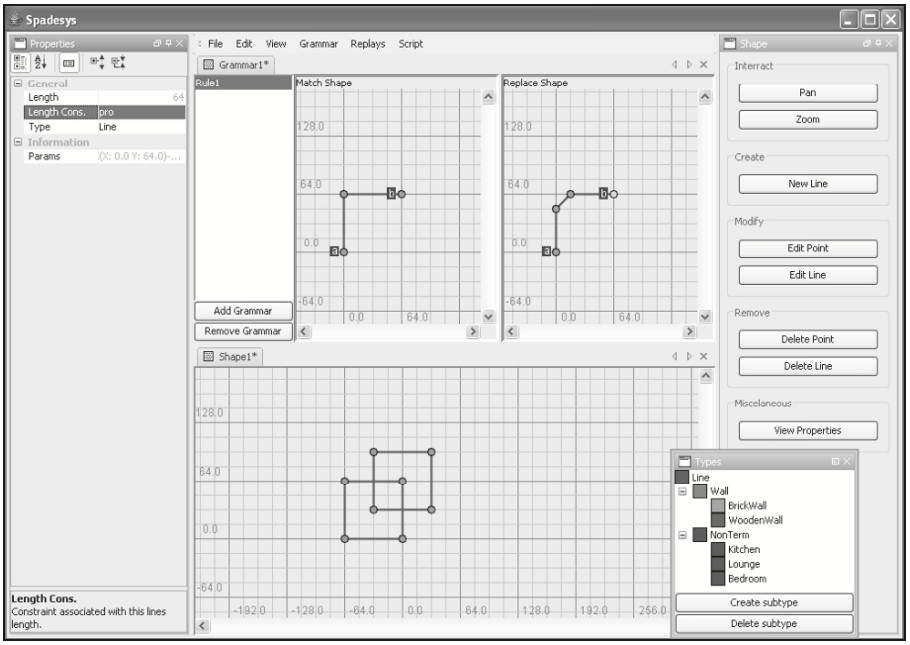


Figure 11. Main Spadesys interface.

The 'Types' window shown in Figure 12 allows the definition of line types as described in extension 3, Section 7. The tree structure models the hierarchical relationships between the types defined. Lines in the match and replace shape can have their type set to any one of the values in this tree.

Each type can have an associated colour, which can be used as the draw colour for every line of that type.

The properties window from Figure 12 dynamically displays the properties of the currently selected object. The editable parameters of lines, points and types etc. can be changed here.

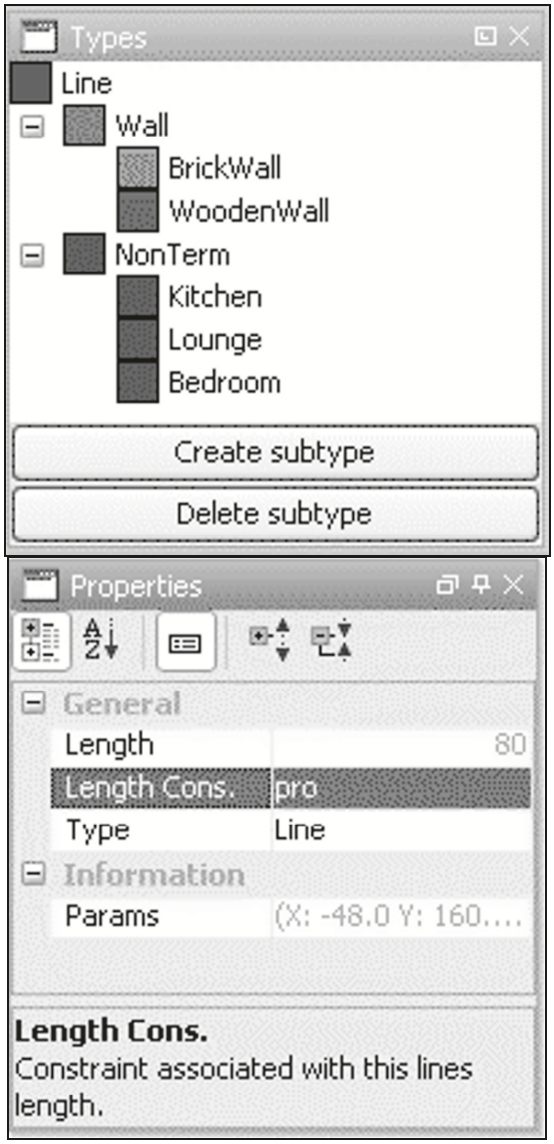


Figure 12. Types and Properties window.

The image from Figure 13 demonstrates that the currently shown grammar rule is the one selected from the list on the left, which contains all the rules in the current grammar set. Each individual rule can be given a name to be able to clearly determine their role.

Figure 14 demonstrates the manual application of a grammar rule to the design shape. The interface presents to the user the location and relative orientation of the match shape as it is found in the design shape. One of the

lines from the match and design shape are dashed, to show their correspondence. The user may cycle through all possible matches, and apply the replace shape at the intended location.

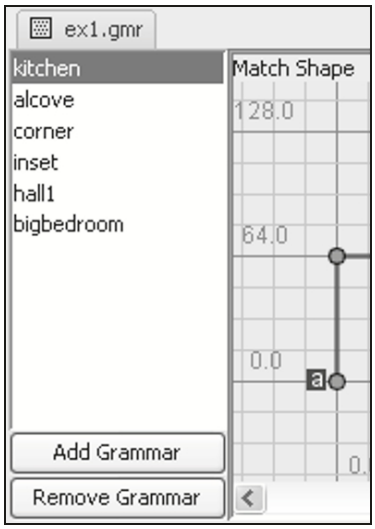


Figure 13. Grammar rule list.

Using the embedded scripting system, the grammar application process can be automated. Various strategies can be used for applying the grammar rules. Simple, random algorithms as well as advanced logical reasoners can be written using the BeanShell scripting language, and executed on a design. The scripting language is similar to the Java language, and allows for rapid development of scripts. The *intelligent designer* itself is implemented as a script. Figure 15 shows the automatic progression of a design, as grammar rules are selected and applied.

10.2. CAD SOFTWARE

Most common modelling operations available in CAD software, such as extrude, bevel, chamfer, slice etc. can be represented using CAD grammars. As en example, the *extrude* operation found in the majority of cad software can be represented using this parametric CAD grammar.

The letters '*a*' and '*b*' in Figure 16 represent tags as described in the modification extension description of this paper. This is to ensure that the connectivity states of the associated points are maintained after the rule is applied. The dashed lines represent the parameterized values; their length should be alterable in some way through the interface. This operation would generally be performed on a polygon face rather than a single edge, in which case the same rule is applied to all edges on a face, rather than a single one.

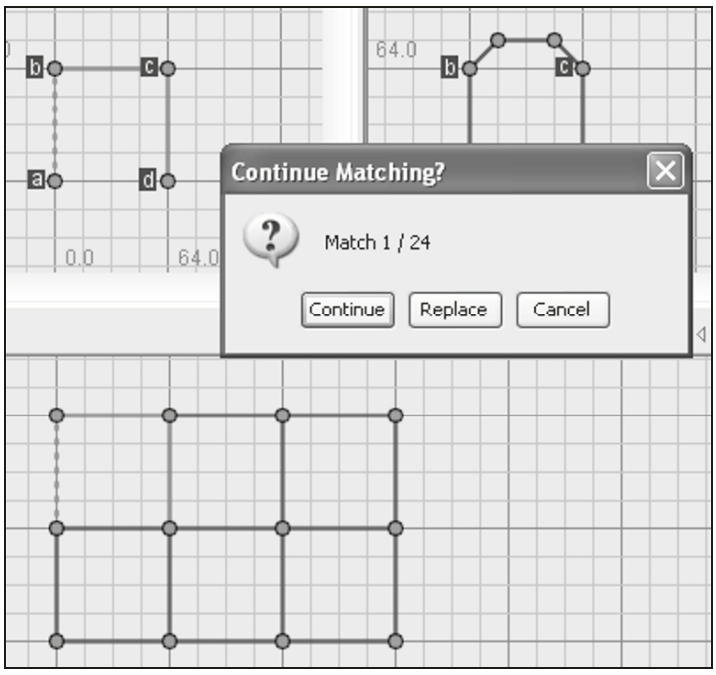


Figure 14. Manual matching and replacement.

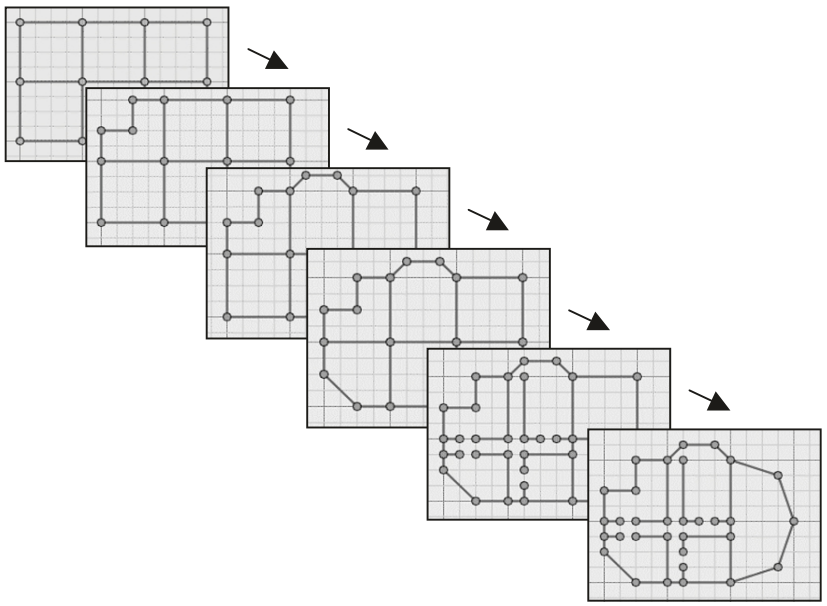


Figure 15. Design progression.

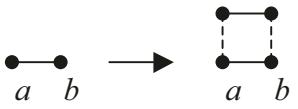


Figure 16. Extrude operation.

Similarly, a 2D bevel operation could be represented as Figure 17.

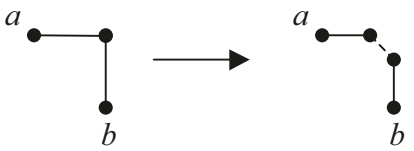


Figure 17. Bevel operation.

11. Conclusion

CAD grammars provide a flexible approach to the applications of shape grammars. The enhanced matching features allow the construction of smaller grammar rule sets. Grammar rules can adapt and match a larger number of relevant configurations due to the length and angle constraint features. The modification features, which allow grammars to directly modify designs introduces a new dimension to design generation.

The emergent features of shape grammars, where large complex designs can be generated from a few simple rules are still present; since traditional grammars can be created that do not take advantage of CAD specific features. The ability to define clear and predictable grammars is also enhanced, as the extended features can be applied where and when desired. Based on a design's requirements, the tradeoff between predictability and emergence can be made by the grammar designer. Similarly, predictable and emergent grammars can be mixed and used together within the same problem.

Using the *line-types* extension has not only functional benefits; in addition grammar rules can become self-documenting, with their features and intentions made clear by the visible type name of every line in the match and replace shapes.

The polymorphic nature of line-types provides more power to the grammar designer, by being able to further specify the intentions of grammars. This allows for the creation of more *'designerly'* grammars, where the design process can flow consistently with the design intention of the grammar designer.

References

Barendsen, E and Smetsers, S: 1999, Graph rewriting aspects of functional programming, *in* H Ehrig, G Engles, HJ Kreowski and G Rozenberg (eds), *Handbook of Graph Grammars and Computing by Graph Transformation*, World Scientific, pp. 62-102.

Knight, TW: 1989, Color grammars: designing with lines and colors, *Environment and Planning B: Planning and Design* **16**: 417-449.

Knight, TW: 1999, Shape grammars: six types, *Environment and Planning B: Planning and Design* **26**: 15-31.

Liew, H: 2004, SGML: A Meta-Language for Shape Grammars, *PhD Dissertation*, Massachusetts Institute of Technology, Cambridge, Mass.

Plump, D: 1999, Term graph rewriting, *in* H Ehrig, G Engles, HJ Kreowski and G Rozenberg (eds), *Handbook of Graph Grammars and Computing by Graph Transformation*, World Scientific, pp. 3-61.

Reffat R: 2002, Utilisation of artificial intelligence concepts and techniques for enriching the quality of architectural design artefacts. *Proceedings of the 1st International Conference in Information Systems* **5**: 1-13.

Stiny, G: 1980, Introduction to Shape and Shape Grammars, *Environment and Planning B* **7**(3): 343-351.

Stiny, G and Mitchell, WJ: 1978, The Palladian grammar, *Environment and Planning B* **5**: 5-18.

Stiny, G: 1992, Weights, *Environment and Planning B: Planning and Design* **19**: 413-430.

COMBINING EVOLUTIONARY ALGORITHMS AND SHAPE GRAMMARS TO GENERATE BRANDED PRODUCT DESIGN

MEI CHOO ANG, HAU HING CHAU, ALISON MCKAY AND ALAN DE PENNINGTON
University of Leeds, United Kingdom

Abstract. Shape grammars have been used to generate new branded product design shapes in accordance with designer preferences in a number of product domains. In parallel, evolutionary algorithms have been established as random search techniques to evolve and optimize designs to meet specific requirements. The research reported in this paper investigated the use of a combined approach, bringing together the shape synthesis capability from shape grammars and the evolution and optimization capability from evolutionary algorithms, to support the generation and evaluation of new product shapes. A system architecture for the integration of shape grammars with evolutionary algorithms is presented. Prototype software based on this architecture is described and demonstrated using a Coca-Cola bottle grammar as a case study.

1. Introduction

A product is an artifact that is manufactured and sold by an enterprise to its customers (Ulrich and Eppinger 2000; Pahl and Beitz 2001). The success of an enterprise depends on its ability to identify customers' needs and create products that meet these needs quickly and at low cost. However, the consumer market is filled with mass-produced products that are virtually indistinguishable from one another. As technologies become mature, customers take the basic performance of products for granted and begin to look for other properties such as price, value, prestige, appearance, brand reputation and convenience. They tend to purchase by brand name rather than technical distinctions (Norman 1998). As a consequence, enterprises have to brand their products distinctively and promote their brands to gain market share. Brand identity becomes an essential strategy to increase competitiveness.

Branded products are delivered to customers through product development processes. A number of different product development processes are proposed in the literature. For example, Ulrich and Eppinger

J.S. Gero (ed.), Design Computing and Cognition '06, 521–539.
© 2006 *Springer. Printed in the Netherlands.*

(2000) divide product development processes into six phases: planning, concept development, system-level design, detail design, testing and refinement, and production ramp-up. Typically enterprises strive to improve their product development processes by producing more designs, more quickly, at lower cost and higher quality. The achievement of these goals enables enterprises to respond better to customer demand. A key to achieve these goals lies in the synthesis of new product shapes that both conform to brand identity and meet specific functional requirements, for example, a given volume for a bottle.

Shape grammar research for product design has focused on the development of product shape or external concept designs and has not stressed the evaluation of the generated designs with respect to functional requirements. Parametric shape grammars have been used to generate branded product design concepts conforming to brand identity but, again, without an explicit relationship to functional requirements. In addition, the sequences of shape grammar rules needed to generate new design concepts have been selected manually.

Evolutionary algorithm research for product design has focused on the evaluation of the generated designs with respect to functional requirements but not on the maintenance of the style or external appearance of products.

The computational approaches of evolutionary algorithms that automatically search and evaluate designs are capable of replacing the manual effort of rule selection and design evaluation needed in the shape grammar design process.

This paper presents the results of research that explored the incorporation of evolutionary algorithms into a shape grammar-based design system. The evolutionary algorithms were used to evaluate generated shapes with respect to a functional requirement. The results of these evaluations were then used to inform the identification of shape grammar rule sequences that were used to create the next generation of shapes. A system architecture for the integration of shape grammars with evolutionary algorithms is presented. Prototype software based on this architecture was built and is demonstrated in this paper using a Coca-Cola bottle grammar as a case study.

2. Shape Grammars

2.1. BACKGROUND

Shape grammars were first introduced by Stiny and Gips in 1972. Shape grammars consist of a set of finite number of shapes **S**, a set of labels **L**, an initial shape **I** and a set of shape rules **R** that define spatial relationships between shapes (Stiny 1980). It is a formal method to generate shapes through a sequence of rule applications beginning from an starting initial shape, **I**. Rules take the forms $A \Rightarrow B$, where A and B are both shapes. As a

demonstration, a simple shape grammar given by Stiny (1976) is used and illustrated in Figure 1. By applying rule 1 twice and rule 2 once on the initial shape, the resulting shape is shown step by step in the Figure 2.

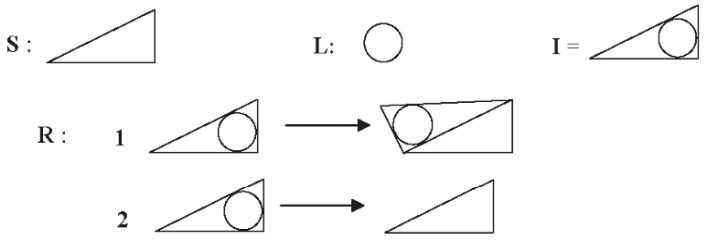


Figure 1. Simple shape grammar.

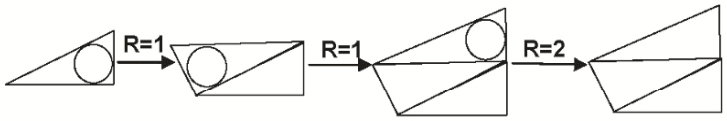


Figure 2. An example pattern generated from the simple shape grammar.

2.2. IMPLEMENTATIONS OF SHAPE GRAMMAR IN GENERATING BRANDED PRODUCT DESIGNS

The visual elements of brand identity can be regarded as an integrated system that includes shapes, colors, and typography/contents (Perry and Wisnom 2002; Wheeler 2003). In the sequence of cognition, the human brain acknowledges and remembers shapes first (Wheeler 2003). Thus, the product shape portrays product identities and gives significant impact to its market share (Perry and Wisnom 2002; Wheeler 2003).

Shape grammars have been used to design the shape of consumer products; the first example in the literature was a coffeemaker grammar (Agarwal and Cagan 1998). The coffeemaker grammar was able to generate four existing branded models of coffeemaker but it did not address the issue of style conformance to one particular brand. It gave similar features among them but not distinct features to allow brand differentiation. The first attempt to capture a brand style using shape grammar was the Dove soap bar grammar (Chau 2002). Other examples are the Harley-Davidson Motorcycle (Pugliese and Cagan, 2002) and Buick automobile grammars (McCormack, Cagan and Vogel 2004), a Cola-Cola bottle grammar (Chau et al. 2004), and a Personal Care Products grammar (Chen et al. 2004). Table 1 gives a comparison of the shape grammars introduced in this paragraph. In most cases, the shape grammars have focused on the development of product shapes and external forms rather than the satisfaction of functional requirements.

TABLE 1. Summary and comparisons of research work in shape grammar for product designs.

Grammar Name	Coffeemaker (1998)	Dove (2002)	Harley-Davidson (2002)	Buick (2004)	Coca-Cola (2004)	Personal Care Products (2004)
Product Scope	Coffee Maker	Soap	Motorcycle	Car	Beverage Bottle	Personal Care Container
Brand Link	Krups, Black & Decker, Proctol Silex, Braun	Dove	Harley-Davidson	Buick	Coca-Cola	Dove, Elvive, Sasson, Gliss Kur, Trevor Sorbie, H & S[1]
Shape/ Geometric Representation in Rules	2D[2]	3D	2D	2D	2D	3D
Number of Rules	100	12	45	63	12	14
Transformation Rule Representation	Components of product	Outline contour of product	Components of product	Components of product	Partitioning of product	Cross-section of product
Essence of Brand Characteristics	1. Heater Unit 2. Filter 3. Base Unit 4. Water Storage Unit 5. Burner Unit	Entire Product	1. 45-degree V-twin engine 2. Teardrop-shaped fuel tank.	1. Grill 2. Hood flow lines 3. Outer hood 4. Fenders 5. Middle hood 6. Emblem	Entire product	Entire product
Generation of New Product	Yes	No[3]	Yes	Yes	Yes	Yes
Shape Generation Criteria	Functional Requirements and Manufacturing Cost	Nil	User Preference (aesthetic)	Designer Preference (aesthetic)	Nil	Nil
Rule Utilisation Method	Manual	Manual	Manual	Manual	Manual	Manual
Distinct Identity of Brand Shape[4]	No	No	Yes	Yes	No	No

[1] H&S – Head & Shoulder.
[2] Although the rules were in 2D, the authors of this grammar showed that it is possible to interpret the resulting shapes in a 3D form.
[3] Dove grammar was not used to generate new Dove shape but was used to generate existing shape of other branded soap.
[4] Distinct identity of Brand Shape is defined as clearly recognisable shape to a particular brand among the user of the product scope.

Shape grammars can be used to generate shapes that conform to brand identity. The generation of such shapes entails a sequence of rule applications. Each rule application involves the selection of rules, identification of sub shapes, implementation of the rule and generation of new shapes. Currently these steps are done manually. This research investigated the use of evolutionary algorithms to perform the rule selection step and determination of parameters automatically while satisfying functional requirement and parameter constraints.

3. Evolutionary Algorithm

3.1. BACKGROUND

There are three main biological evolutionary systems that constitute evolutionary algorithms. These three main biological evolutionary algorithms are: evolution strategies, evolutionary programming and genetic algorithms (Whitley 2001). These biological evolutionary systems were introduced independently by several computer scientists in the 1950s and 1960. Evolution strategies were introduced by Ingo Rechenberg in Germany in 1960s and were further developed by Jams-Paul Schwefel. Evolutionary programming was introduced by Fogel, Owens, and Walsh in 1966. Genetic algorithms were introduced by John Holland in the 1960s and developed further by Holland himself together with his students and colleagues at the University of Michigan (Mitchell 1997). These pioneers shared the same idea that the evolution process could be simulated and used as an optimization tool for engineering problems. The general approach in all these systems was to evolve a population of candidate solutions to a given problem using operators inspired by natural genetic variation and natural selection. Since these inventions, there has been widespread of interactions among researchers and evolutionary algorithms have been extensively applied to solve many engineering problems.

Terminologies described in evolutionary algorithms are normally analogous to their genetic counterparts in biology. An individual is an encoded solution to some problem. Typically, an individual solution is represented as a string (or string of strings) corresponding to a biological genotype. This genotype defines an individual organism when it is expressed (decoded) into a phenotype. The genotype is composed of one or more chromosomes, where each chromosome is composed of separate genes which take on certain values (alleles). A locus identifies a gene's position within the chromosome. This evolutionary algorithms terminology is summarised in Table 2. A set of genotypes is termed as a population. Three major evolutionary operators operate on an evolutionary algorithm's population. These major evolutionary operators are recombination, mutation, and selection. In general terms, recombination exchanges genetic material between a pair of parents' chromosomes. Mutation flips (replaces) a symbol

at a randomly chosen locus with a randomly chosen new symbol. Mutation does not happen on every individual; it is executed whenever the mutation probability of an individual is higher than mutation rate. The selection gives individuals with higher fitness a higher probability of contributing one or more offspring in the succeeding generation. The processes of recombination, mutation and selection for reproduction continue until some conditions are met (for example, it reaches the maximum generation). An evolutionary algorithm requires both an objective and a fitness function. The objective function defines the evolutionary algorithm's optimal condition in the problem domain. On the other hand, the fitness function (in the algorithm domain) measures how 'well' a particular solution satisfies that condition and assigns a real-value to that solution.

TABLE 2. Explanation of evolutionary algorithms terms.

Evolutionary Algorithms	**Explanation**
Chromosome (string, individual)	Solution (coding), part of a complete genotype
Genes (bits)	Part of solution
Locus	Position of gene
Alleles	Values of gene
Phenotype	Decoded solution
Genotype	Encoded solution

Historically, evolutionary algorithms have been used for functional optimization, control and machine learning (Goldberg 1989). As such, initial applications of evolutionary algorithms in design were largely focused on the optimization of design parameters. However, more recent research in evolutionary algorithms has been related to the generation of forms or creative designs (Rosenman 1997; Bentley et al. 2001; Renner and Ekárt 2003). Integration of evolutionary algorithm approaches and shape grammars has been attempted in architectural design (Chouchoulas 2003), structural design (Gero et al. 1994) and product design (Lee and Tang 2004). Their work used genetic algorithms to explore design possibilities and shape grammar to provide a syntactic generation method. Chouchoulas used genetic algorithms and shape grammars to evolve architectural layouts. In his work, he generated room layout designs for high rise building that were evaluated against a number of functional requirements (Chouchoulas 2003). He applied simple rectangular oblongs to represent abstract internal room organizations on different floors which required further refinements to complete a building layout. His work was not linked to any existing architectural style. Gero et al. (1994) has used genetic algorithms to produce new generations (by evolution) of a beam structure shape grammar starting from an initial shape grammar. They showed that the evolved shape grammar was able to produce better beam structures than the initial shape grammar. The performance of the evolved shape grammar in each generation was ranked computationally by comparing two conflicting

physical properties (maximise moment of inertia, minimise beam section perimeter) in the shapes that were generated from the grammar. Lee has also used genetic algorithms to evolve shape grammars for particular types of consumer product; shapes generated from the evolved shape grammars were evaluated manually based on human preference by looking at the designs generated in each generation. Evolutionary algorithms have been successfully applied to many real world problems. However, existing applications have shown that standard evolutionary algorithm approaches alone are not able to achieve the desired results; customization of evolutionary algorithms, by the incorporation of domain specific measures, is needed (Nagendra et al. 1996). The evolutionary algorithm used in the research presented in this paper was customised to enable the evaluation of alternative bottle shapes with respect to their volume.

The brief review above discussed about existing evolutionary algorithms that have been integrated with shape grammar. Another important integration system that combines optimisation techniques and shape grammar is shape annealing introduced by Cagan and Mitchell in 1993. Shape annealing is integration between simulated annealing and shape grammar. The search process in simulated annealing is different from evolutionary algorithms in that it borrows ideas from physical processes rather than biology. The use of the shape annealing approach was shown in geodesic dome style designs (Shea and Cagan 1997) and truss designs (Shea and Cagan 1999). In the application of truss design, Shea and Cagan (1999), have used specific grammar rules to generate golden triangles; this has allowed truss structures reflecting style of golden proportions to be built.

4. Integrating Evolutionary Algorithms and Shape Grammars to Generate Branded Product Design to Meet Functional Requirement

Two main integration interfaces are used to evolve branded product designs: the encoding and decoding interfaces. The encoding interface uses shape grammar rules, initial shapes, parameters and constraints to provide the blueprint for the genotype to be used by the evolutionary algorithm. Each shape rule has its associated shapes, parameters and constraints. The decoding interface allows the evolutionary algorithm to decode the genotype into a phenotype. The phenotype is the actual representation of shape rules and parameters needed to generate the actual design shapes. This encoding process is needed during the early planning stage of evolutionary algorithms. Genotype is coded into a 2-dimensional array of data structure and in the context of shape grammar rules and parameters. As an example, a genetic representation is explained using a case study in Section 5. The decoding interface has two purposes: 1) it allows the evolutionary algorithm to evaluate and rank the phenotype performance with respect to functional requirements during the fitness assignment process; 2) information from it allows final shape to be produced using shape grammar implementation

software. A system architecture for the integration of shape grammars with evolutionary algorithms is given in Figure 3.

5. A Case Study On A 2-D Coca-Cola Bottle Grammar

This case study demonstrated the application of the integrated architecture shown in Figure 3 to produce some viable product shapes that conformed to the Coca-Cola brand identity. The evolutionary algorithm was specifically developed by combining customised evolutionary algorithm sub-functions: recombination, mutation and selection procedures. The representation was a combination of rule numbers (reference number in integers) and their associated parameters (floating point numbers). The case study was based on Coca-cola bottle shape grammar as shown in Figure 4 (Chau et al. 2004).

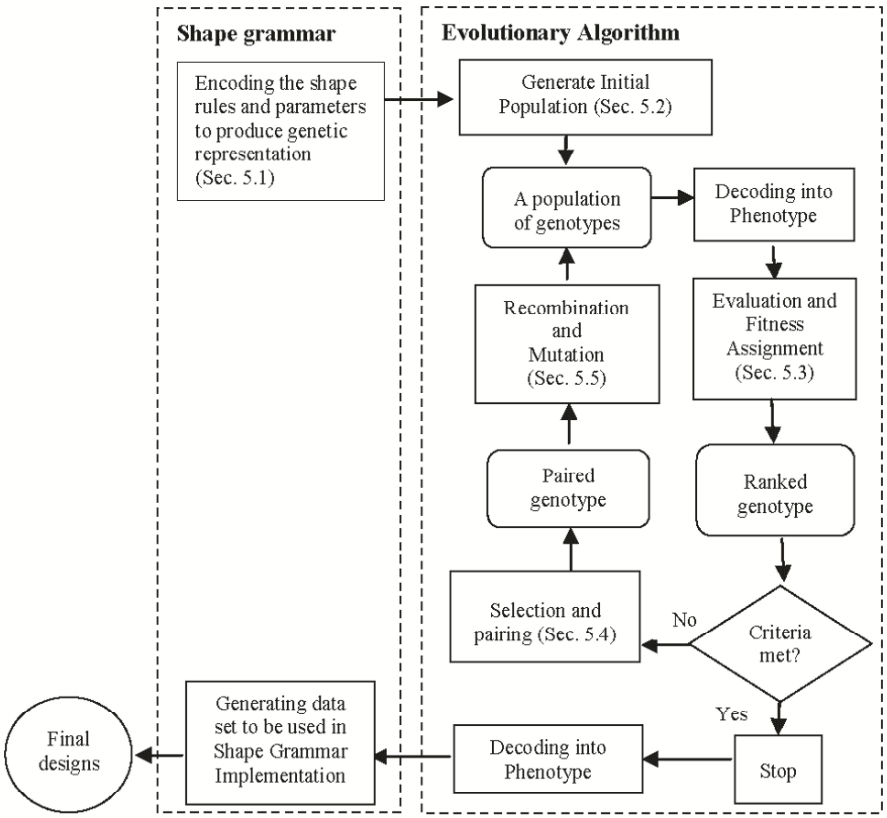


Figure 3. An evolutionary algorithm and shape grammar integration architecture.

The Coca-Cola bottle shape grammar provides information on specific bottle sections, characteristics of the shape and contour in each section, and the relationships between sections. There are no specific rules to generate values for different diameters and heights. There are also no specific constraints on diameters and heights in order to maintain brand image, it is

still an open issues as to how to maintain brand image. In this application, each bottle section is described using diameters and heights. Diameters and heights provide the start point and end point of each connecting curve. Curves in each bottle section are formed by three points. The curves are determined manually after completing the evolutionary process by approximating the curve shape used in the shape grammar rules.

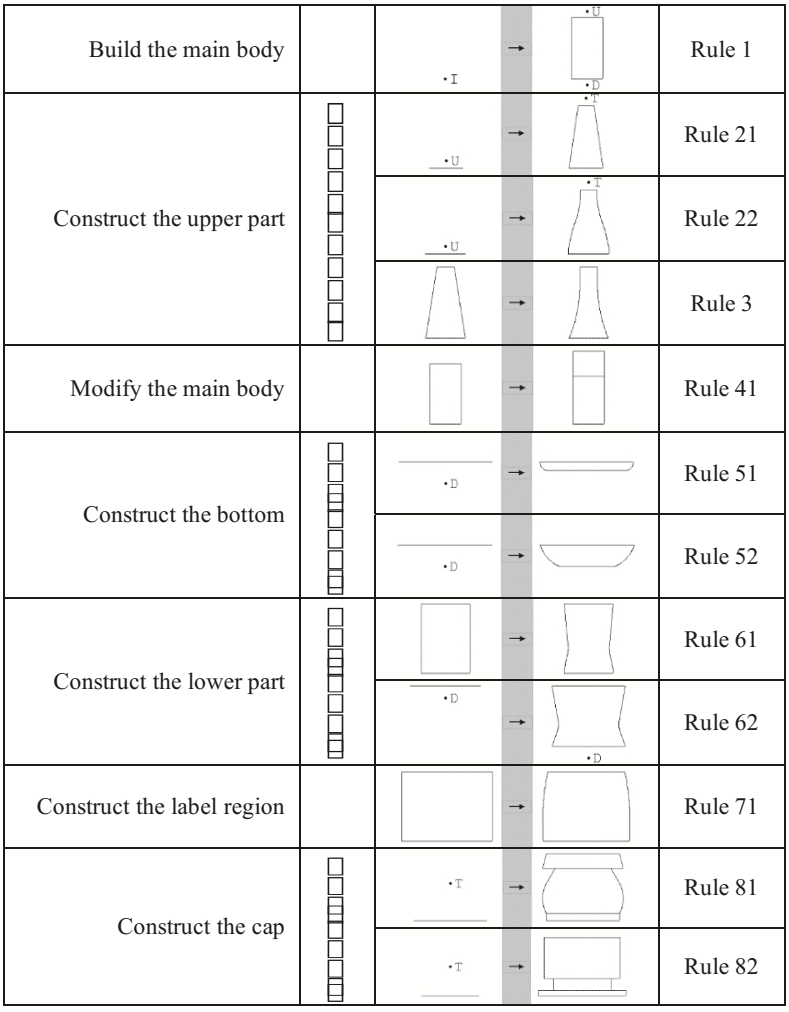


Figure 4. Shape rules for the Coca-Cola bottle grammar, reproduced from Chau et al. (2004).

In the implementation, the diameters and heights of each section are set within finite ranges ([minwidth, maxwidth] and [minheight, maxheight]), and currently their values are minwidth = minheight = 0 cm and maxwidth =

maxheight = 10 cm. These parameters were incorporated to facilitate the calculation of the volume (functional requirement) for the bottle shapes produced by the prototype system. The use of volume was a demonstration of one possible application of evolutionary algorithms to generate product designs that both conform to a style captured in shape grammar rules and meet a given functional requirement.

5.1. GENETIC REPRESENTATION

Five sections or parts are used to define the Coca-Cola bottle: cap, upper part, label region, lower part and bottom, Figure 5. There are a total of seven rule groups in the Coca-Cola bottle grammar, Figure 4. Starting from the rules for building the main body, there are other rules for construction of the upper part, modification of the main body, construction of the bottom, construction of the lower part, construction of the label region and construction of the cap. A rule group may contain more than one rule, for example, the construction of the upper part contains three separate rules that produce different shapes on top of the main body.

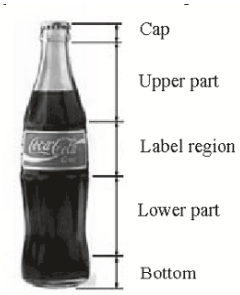


Figure 5. Graphical illustration of Coca-Cola bottle reproduced from (Chau et al. 2004).

Additional parameters are used to describe the height and diameters of each bottle section (refer Figure 6). For example, three parameters are used to describe the upper part section: bottom diameter (Dia $\Phi_{1,2}$), top diameter (Dia $\Phi_{2,2}$) and height ($Height_{3,2}$).With these additional parameters for diameters and heights in each bottle part, a genetic representation was built to represent the genotype to encode these seven rule groups and their parameters. The genotype (genetic representation) resembled a (m x n) matrix. In this case study, a 5x7 matrix is used and illustrated in Figure 7.

The construction sections and associated rule numbers are given in Table 3. Based on the shape rules of the Coca-Cola bottle grammar, there is more than one rule in each rule group but only one rule can be selected to be executed in a given computation step. In the construction of a bottle, the use of a rule from groups RG1, RG2, RG3 and RG4 is compulsory to produce a valid bottle design because every bottle must have a body, an upper part, a bottom and a cap. The rules in groups RG5, RG6 and RG7 are executed to produce variation in the bottle designs and their use is optional.

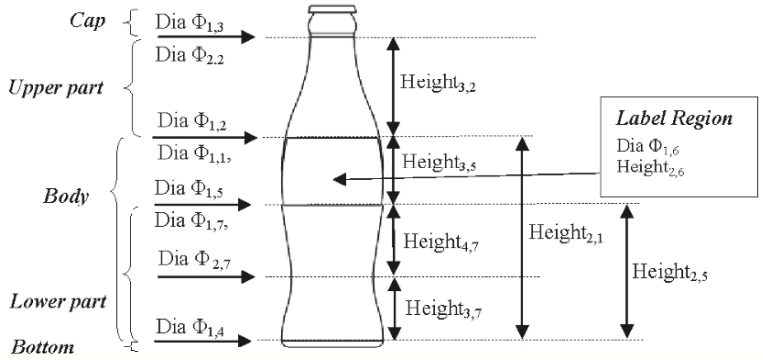


Figure 6. Diameters and heights of the bottle parts.

The structures of bottle sections in each rule group vary, Table 3, and they were generalised to facilitate the calculation of the total volume in that the curvilinear parts were simplified into linear lines. Thus, the volume calculated was an approximation to the actual volume. The shape of the bottom and cap were not included in the volume calculations.

RG1	RG2	RG3	RG4	RG5	RG6	RG7
Dia $\Phi_{1,1}$	Dia $\Phi_{1,2}$	Dia $\Phi_{1,3}$	Dia $\Phi_{1,4}$	Dia $\Phi_{1,5}$	Dia $\Phi_{1,6}$	Dia $\Phi_{1,7}$
Height$_{2,1}$	Dia $\Phi_{2,2}$			Height$_{2,5}$	Height$_{2,6}$	Dia $\Phi_{2,7}$
	Height$_{3,2}$			Height$_{3,5}$		Height$_{3,7}$
						Height$_{4,7}$

Figure 7. Genotype of the Coca-Cola bottles.

TABLE 3. Properties in each rule group.

Rule Group (RG)	Construction part	Rules	Structure
RG1	Main body	1	Cylinder
RG2	Upper bottle part	21, 22, 3	Frustum
RG3	Bottle bottom	51, 52	
RG4	Bottle cap	81, 82	
RG5	Modification on the main body	41	Cylinder
RG6	Label region	71	Cylinder
RG7	Lower part	61, 62	Two frustums

The corresponding parameters in each rule group depend on the resulting shape when one of the rules in a rule group is applied. In this particular case study, an evolutionary algorithm was used to generate rule sequences and associated parameters to achieve the total volume of 500ml. Each body part

had its own parameters: diameters and heights. Both the body part diameters and the body part heights were set in the range [minwidth, maxwidth] and [minheight, maxheight].

The initial population was generated randomly. The probability of selecting a given rule in a given rule group was equal; there was no bias to any particular rule. For example, in rule group two (RG2), there were three rules that could have been selected: 21, 22, and 3, the probability of each being selected was equal, and the probability for each rule was 1/3. The parameters generated were also random and the diameters of the connecting bottle parts were made equal to ensure that the bottle designs were valid.

5.2. EVALUATION AND FITNESS ASSIGNMENT

The objective function for this particular case study was to minimize the difference between the bottle volume () and a desired target volume (). Mathematically, it can be written as equation (1) which is equivalent to equation (2).

$$\text{Minimise} \quad f(\nu) = -(\nu - \nu_{target})^2 \tag{1}$$

$$\begin{aligned} \text{Maximise } g(\nu) &= -f(\nu) + C \\ &= -(\nu - \nu_{target})^2 + C \end{aligned} \tag{2}$$

A constant C was added to g(v) to ensure that the objective function took only positive values in its domain (Michalewicz 1996). The volume of a bottle, v, refers to the total volume that the bottle can contain. The volume calculation does not consider the cap and bottom as these do not normally contain the content of the bottle. Each bottle had four possible body parts to be included in the volume calculation (refer section 5.1). The volume of each body part was summed to obtain the total volume of a bottle. The structures of the body parts were varied and could be either a cylinder or a frustum. Individuals in every generation were evaluated based on the volume of the bottle in their phenotype. Fitness assignment is the process of assigning selection probability to each of the individuals in the population. The selection probability is calculated using equation (3).

$$p_i = g(v_i) / F \tag{3}$$

Where p_i is the selection probability for individual i; v_i is the total volume for individual i; and F is the total fitness of the population, given by equation (4).

$$F = \sum_{i=1}^{N} g(v_i) \tag{4}$$

This method is also known as the proportional fitness assignment approach (Goldberg 1989; Michalewicz 1996).

5.3. SELECTION AND PAIRING

The selection procedure was based on stochastic universal sampling (Baker 1987). It provided zero bias and minimum spread. In the procedure (Pohlheim 2005), the individuals were mapped to contiguous segments of a line where each individual's segment was proportional in size to its selection probability. Some equally spaced pointers are placed over the line as many as there are individuals to be selected. If the number of individuals to be selected is N, then the spacing between the pointers is $1/N$. The position of the first pointer is given by a randomly generated number in the range [0, $1/N$]. Table 4 shows an example of 10 individuals and their corresponding selection probability. If six individuals are to be selected from a population of ten individuals, then N = 6 and pointer spacing, 1/N = 0.167. A random number are generated from the range of [0, 0.167]. As shown in Figure 8, using the index (from 1 to 10), the individuals to be selected in the example are 1, 2, 3, 4, 6, and 8. These individual will later be paired up to undergo the reproduction process in recombination and mutation operations.

TABLE 4. Selection probability.

Individual index	1	2	3	4	5	6	7	8	9	10
Selection probability	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02

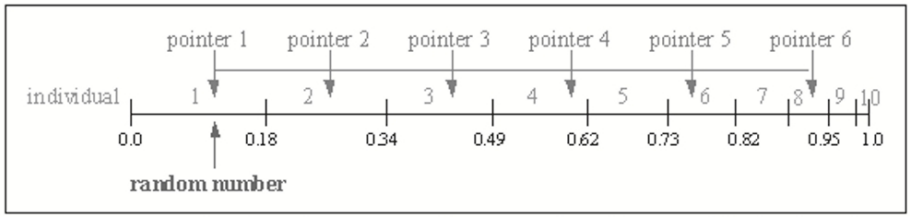


Figure 8. Stochastic universal sampling, reproduced from Pohlheim (2005).

5.4. GENETIC OPERATIONS

There were one genetic recombination and one mutation operation in this case study. The recombination operation was a modification of the single point crossover (explained in Section 5.5.1) whereas the mutation operation could operate on both rules and parameters associated with the rules (explained in Section 5.5.2).

5.4.1. Recombination

The recombination operation began with the random selection of a pair of parents who were chosen for reproduction. Then, a crossover point, represented by an integer, was randomly generated in the range [1, m-1]. The crossover point was the starting location where genetic material between parents was swapped. Figure 9 shows a single-point crossover operating on Parent1 and Parent2; each parent is cut and recombined with a piece of the other. The crossover operation involved in this case study was a modification on this single-point crossover and is illustrated in Figure 10. Two parents P1 and P2 were selected and a crossover point was located in a position equal to 2. This implied that rule numbers in RG1 and RG2 were maintained in the same positions of P1 and P2, but rule numbers RG3 to RG7 were swapped. The resulting chromosomes of the offspring: C1 and C2 are also shown in Figure 10. After the crossover operation, the diameters of the adjacent body parts were usually different. In order to produce a smooth transition between body parts, their diameters were averaged to obtain a new diameter value.

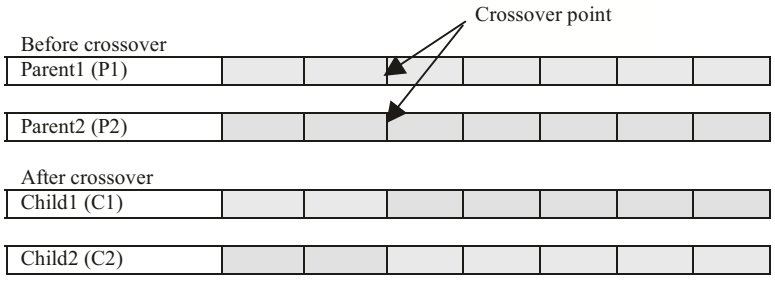


Figure 9. Single-point crossover.

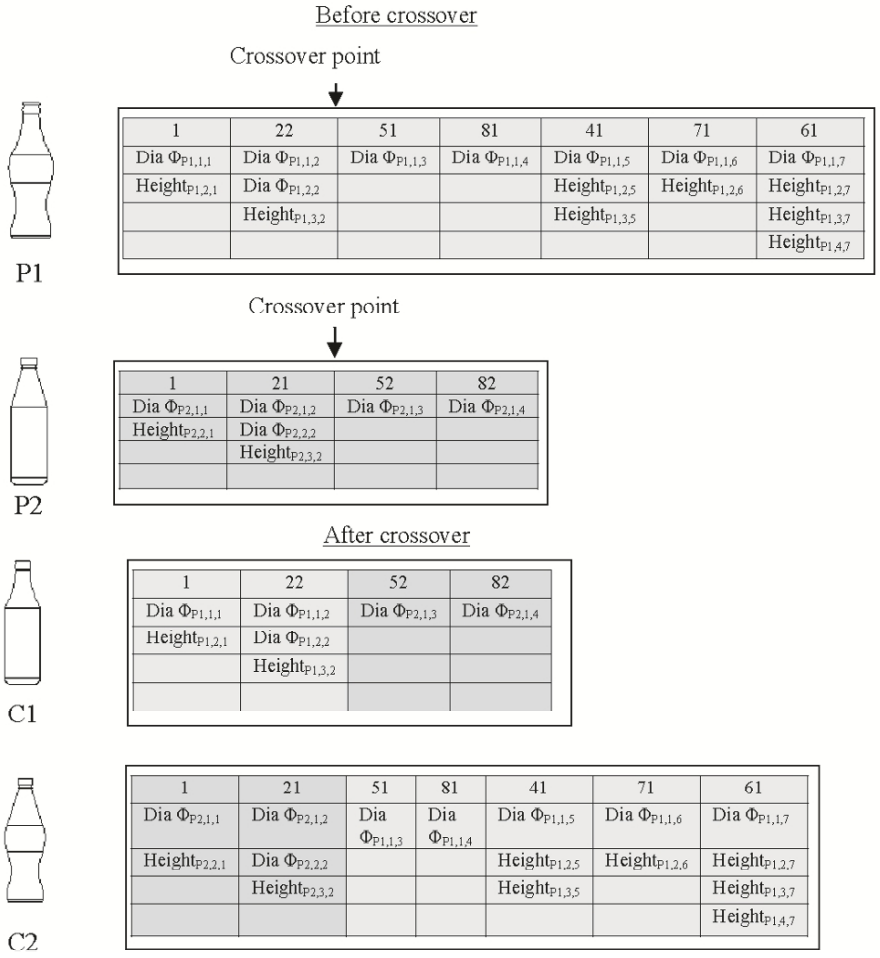
5.4.2. Mutation

The mutation operator could change any chosen rule into a different rule in the same group. For example, rule 21 could become rule 22 or 3. The associated rule parameters could be altered into different values. The rule parameters were real numbers, and they were randomly mutated in a predefined range as described in Section 5.1. The mutation location was an integer, chosen randomly in the range [1, m-1]. The mutation operation was executed with a probability equal to the mutation rate.

6. Experimental Results

The case study was coded and implemented following the evolutionary algorithm and shape grammar integration architecture given in Figure 3. A display of the program output is shown in Figure 11. The results shown in the program output can be viewed by the user by scrolling to view the other

near best solutions in the list of top ten solutions. The last results displayed by the output were the best solutions as shown in Figure 11.



1	22	51	81	41	71	61
Dia $\Phi_{P1,1,1}$	Dia $\Phi_{P1,1,2}$	Dia $\Phi_{P1,1,3}$	Dia $\Phi_{P1,1,4}$	Dia $\Phi_{P1,1,5}$	Dia $\Phi_{P1,1,6}$	Dia $\Phi_{P1,1,7}$
$Height_{P1,2,1}$	Dia $\Phi_{P1,2,2}$			$Height_{P1,2,5}$	$Height_{P1,2,6}$	$Height_{P1,2,7}$
	$Height_{P1,3,2}$			$Height_{P1,3,5}$		$Height_{P1,3,7}$
						$Height_{P1,4,7}$

1	21	52	82
Dia $\Phi_{P2,1,1}$	Dia $\Phi_{P2,1,2}$	Dia $\Phi_{P2,1,3}$	Dia $\Phi_{P2,1,4}$
$Height_{P2,2,1}$	Dia $\Phi_{P2,2,2}$		
	$Height_{P2,3,2}$		

1	22	52	82
Dia $\Phi_{P1,1,1}$	Dia $\Phi_{P1,1,2}$	Dia $\Phi_{P2,1,3}$	Dia $\Phi_{P2,1,4}$
$Height_{P1,2,1}$	Dia $\Phi_{P1,2,2}$		
	$Height_{P1,3,2}$		

1	21	51	81	41	71	61
Dia $\Phi_{P2,1,1}$	Dia $\Phi_{P2,1,2}$	Dia $\Phi_{P1,1,3}$	Dia $\Phi_{P1,1,4}$	Dia $\Phi_{P1,1,5}$	Dia $\Phi_{P1,1,6}$	Dia $\Phi_{P1,,1,7}$
$Height_{P2,2,1}$	Dia $\Phi_{P2,2,2}$			$Height_{P1,2,5}$	$Height_{P1,2,6}$	$Height_{P1,2,7}$
	$Height_{P2,3,2}$			$Height_{P1,3,5}$		$Height_{P1,3,7}$
						$Height_{P1,4,7}$

Figure 10. Recombination operations.

The best solution found after 1000 generations for a population size of 50 individuals, crossover rate 0.5 and mutation rate 0.5, is highlighted in the Figure 11. The best solution shows that the evolutionary algorithm found a bottle volume of 499.92 ml.

Based on the shape rules and associated parameters; the bottle shape was generated using the Coca-Cola bottle grammar implementation software developed by Chau (Chau et al. 2004). The resulting shape is shown in Figure 12.

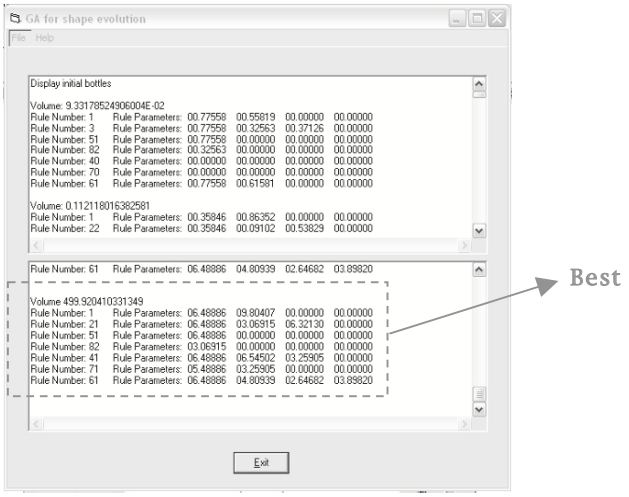


Figure 11. The output interface in the implementation and the best solution (in cm).

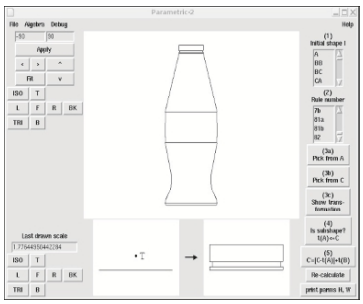


Figure 12. The best solution modelled using Coca-Cola shape grammar implementation.

The resulting shape has a similar style to the Coca-Cola contour bottle in that it has an upper part which follows an earlier style of Coca-Cola bottle, as well as a standard label region and a lower part that imitates the well-known Coca-Cola contour section.

It is possible to generate different solutions with volumes within a small tolerance that have different forms. Figure 13 shows the results of best solutions as the generation number increases. These results are obtained by setting the population size to be 100, recombination rate 0.5 and mutation rate 0.7. The details of rule sequences and associated parameters of selected results are given in Table 5. These details show that the different volumes have different rule sequences and associated parameters and would therefore give different forms if modelled graphically using Coca-Cola shape grammar implementation.

TABLE 5. Detailed results of bottle volumes, rule sequences and associated parameters.

Population size = 100 Crossover rate = 0.5 Mutation rate = 0.7			
Generation number	Volume	Rule sequence	Rule and parameters
50	507.5007	1 21 52 82	1 07.91666 07.72321 21 07.91666 03.25223 04.91313 52 07.91666 82 03.25223
100	503.1757	1 3 52 82 41 61	1 06.38506 09.14664 3 06.38506 01.66892 05.50197 52 06.38506 82 01.66892 41 06.38506 07.89625 01.25040 61 06.38506 04.07032 02.90657 04.98967
150	497.7858	1 21 3 52 82 41 71 61	1 06.41059 09.18485 3 06.41059 02.16005 05.65698 52 06.41059 82 02.16005 41 06.41059 07.55557 01.62928 71 06.29811 01.62928 61 06.41059 04.06910 03.19670 04.35887
500	500.8721	1 22 52 82 41 71 61	1 06.20483 09.65275 22 06.20483 02.18049 06.35109 52 06.20483 82 02.18049 41 06.20483 07.81973 01.83302 71 06.17222 01.83302 61 06.20483 04.25747 04.33448 03.48525
1000	500.0054	1 21 52 82 41 71 61	1 06.00181 07.83138 21 06.00181 02.69377 05.80848 52 06.00181 82 02.69377 41 06.00181 07.98335 01.58121 71 05.76209 01.58121 61 06.00181 04.80297 03.53589 04.44746

7. Discussions and Conclusions

The research reported in this paper has demonstrated that evolutionary algorithms can be used to generate a number of shape grammar rule sequences and associated parameters automatically and the designs can be evaluated with respect to a single functional requirement (volume of the bottle). The case study showed that it is possible to integrate evolutionary algorithms and shape grammars to deliver product shapes with a particular style and that meet functional requirements. This work can be further expanded to investigate and compare results with other solution from same-brand and other competing brands. This case study is only a starting point to investigate issues in the design of branded product. Considerations of other quantifiable evaluation criteria are currently underway.

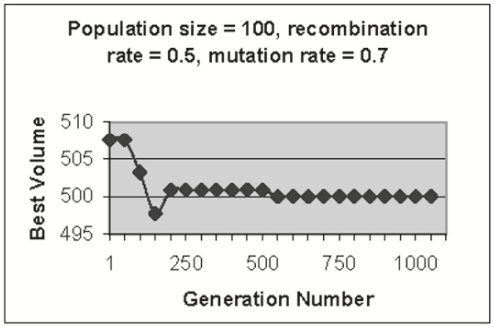


Figure 13. The best solutions converging to 500 ml when generation number increases.

Acknowledgements

The authors would like to thank Ms XiaoJuan Chen for her insightful comments and assistance on the reported research. The authors would also like to express their appreciation to Ministry of Science, Technology and Innovation of Malaysia and Universiti Kebangsaan Malaysia for their scholarship and financial support.

References

Agarwal, M and Cagan, J: 1998, A blend of different tastes: the language of coffeemakers, *Environment and Planning B: Planning and Design* **25**(2): 205-226.

Baker, JE: 1987, Reducing bias and inefficiency in the selection algorithm, *in* JJ Grefenstette (ed), *International Conference on Genetic Algorithms and their Applications*, Hillsdale, New Jersey, USA, Lawrence Erlbaum Associates, pp. 14-21.

Bentley, PJ, Gordon, T, Kim, J and Kuma, S: 2001, New trends in evolutionary computation. *Congress on Evolutionary Computation*, Seoul, Korea.

Cagan, J and Mitchell, WJ: 1993, Optimally directed shape generation by simulated annealing, *Environment and Planning B: Planning and Design* **20**: 5-12.

Chau, HH: 2002, *Preserving Brand Identity in Engineering Design Using a Grammatical Approach*, Leeds University.

Chau, HH, Chen, X, McKay, A and de Pennington, A: 2004, Evaluation of a 3D shape grammar implementation, *in* JS Gero (ed), *Design Computing and Cognition '04*, Kluwer, Dordrecht, pp. 357-376.

Chen, X, McKay, A, de Pennington, A, and Chau, HH: 2004, Package shape design principles to support brand identity, *14th IAPRI World Conference on Packaging*, Stockholm, Sweden.

Chouchoulas, O: 2003, *Shape Evolution: An Algorithmic Method for Conceptual Architectural Design Combining Shape Grammars and Genetic Algorithms*, Department of Architectural and Civil Engineering, University of Bath.

Goldberg, DE: 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, London.

Gero, JS, Louis, J and Kundu, S: 1994, Evolutionary learning of novel grammars for design improvement, *AIEDAM* **8**(2): 83-94.

Lee, HC and Tang, MX: 2004, Evolutionary shape grammars for product design, *7th International Conference on Generative Art*, Politecnico di Milano University.

McCormack, JP, Cagan, J and Vogel, CM: 2004, Speaking the Buick language: Capturing, understanding, and exploring brand identity with a shape grammar, *Design Studies* **25**(1): 1-29.

Michalewicz., Z: 1996, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, London.

Mitchell, M: 1997, *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, Massachusetts, London.

Nagendra, S, Jestin, D, Gürdal, Z, Haftka, RT and Watson, LT: 1996, Improved genetic algorithm for the design of stiffened composite panels, *Computers and Structures* **58**(3): 543-555.

Norman, DA: 1998, *The Invisible Computer: Why Good Products can Fail, The Personal Computer is so Complex, and Information Appliances are the Solution*, MIT Press, Cambridge, Massachusetts.

Pahl, G and Beitz W: 2001, *Engieering Design: A Systematic Appraoch,* Springer-Verlag, London.

Perry, A and Wisnom D: 2002, *Before the Brand: Creating the Unique DNA of an Enduring Brand Identity*, McGraw-Hill.

Pohlheim, H: 2005, *GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab*, (2000-2005), version 3.5, *www.geatbx.com*.

Pugliese, MJ and Cagan J: 2002. Capturing a rebel: Modeling the Harley-Davidson brand through a motorcycle shape grammar, *Research in Engineering Design* **13**(3): 139-156.

Renner, G and Ekárt A: 2003, Genetic algorithms in computer aided design, *Computer Aided Design* **35**(8): 709-726.

Rosenman, MA: 1997, An exploration into evolutionary models for non-routine design, *Artificial Intelligence in Engineering* **11**(3): 287-293.

Shea, K and Cagan, J: 1997, Innovative dome design: Applying geodesic patterns with shape annealing, *AIEDAM* **11**: 379-394.

Shea, K and Cagan, J: 1999, Languages and semantics of grammatical discrete structures, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **13**: 241-251.

Stiny, G and Gips, J: 1972, Shape grammars and the generative specification of painting and sculpture, *in* CV Freiman (ed), *Information Processing 71*: *Proceedings of IFIP Congress*, Amsterdam, North Holland, pp. 1460-1465.

Stiny, G: 1976, Two exercises in formal composition, *Environment and Planning B: Planning and Design* **3**: 187-210.

Stiny, G: 1980, Introduction to shape and shape grammar, *Environment and Planning B: Planning and Design* **7**: 399-408.

Ulrich, KT and Eppinger SD: 2000, *Product Design and Development*, Irwin/McGraw-Hill, Boston.

Wheeler, A.: 2003, *Designing Brand Identity: A Complete Guide to Creating, Building, and Maintaining Strong Brands*, John Wiley, USA.

Whitley, D: 2001, An overview of evolutionary algorithms: Practical issues and common pitfalls, *Information and Software Technology* **43**(14): 817-831.

A SEMANTIC VALIDATION SCHEME FOR GRAPH-BASED ENGINEERING DESIGN GRAMMARS

STEPHAN RUDOLPH
University of Stuttgart, Germany

Abstract. Grammars have been used for the generation of various product designs (e.g. coffeemakers, transmission towers, etc.). Like any other formal language in computer science, the correctness of the designs generated during the translation process possesses the three distinct aspects of *syntax*, *semantic* and *pragmatic*. While compilers can check the syntax and can guarantee the pragmatic correctness by constraint processing, the core question of the semantic correctness was still unresolved. In this work the semantic correctness is achieved through the introduction of the concept of a semantic hull for the language vocabulary. This allows to establish a semantic validation scheme for graph-based engineering design grammars despite the fact that computers are *per se* incapable of processing semantics.

1. Introduction

Design grammars as a means of formal design synthesis methods have already been applied successfully in the automated generation of a variety of engineering, architectural and biological designs (e.g. gears, coffeemakers, houses, transmission towers, plants, etc.). Design grammars consist of the vocabulary, a rule set and form together with an axiom (i.e. the starting symbol for the translation process) a production system. The correctness of the language expressions generated during the translation process possesses the three distinct aspects of syntax, semantic and pragmatic like any other formal language definition in computer science.

Since the expression syntax may be automatically checked by a compiler and the pragmatic correctness of the designs may be guaranteed by a constraint processing mechanism, they do not represent an issue in the validation. The aspect of the semantic correctness of the generated language expressions however has been still open and represents up to date a major bottleneck in the validation of automated engineering design generation.

Design languages based on grammar definitions have already a long history in automatic design generation. Since the early works on generative

J.S. Gero (ed.), Design Computing and Cognition '06, 541–560.
© 2006 *Springer. Printed in the Netherlands.*

design (Stiny 1980) a long list of design generation in a variety of domains such as architecture, biology, mechanical and aeronautical engineering have appeared, see the overview book by Antonsson and Cagan (2001). However, while automatic design generation seems to be an accepted procedure to create new designs, little if not no attention at all has been given until now to the problem of design verification in these domains.

As an important exception to this statement, the design of very large scale integration (VLSI) chips needs to be mentioned. The automation of the VLSI chip design (Horstmannshoff 2002) is based on a functional description of the circuits' functional properties and can be achieved nowadays at the press of a button by means of powerful design compilers (Synopsis 2004).

However, this design automation was only possible because of the fact that the domain of VLSI chip design is theoretically understood and reliable simulation codes do exist for its verification. From the viewpoint of formal languages this enormous progress in design automation was only feasible because of the complete theoretical coverage of all relevant design aspects in this field and their successful projection onto syntactic operations.

In order to prepare the grounds for the semantic validation scheme for graph-based engineering design grammars, the current spectrum of formal language applications to design generation is illustrated here ranging from *string-based* design languages for plant generation over *shape-based* design languages for coffeemaker generation and transmission tower optimization to *graph-based* languages for conceptual satellite design.

1.1. ELEMENTARY DEFINITIONS

Since the terms of *grammar*, *language*, *syntax*, *semantic* and *pragmatic* will be used throughout the paper, they are illustrated in Figure 1 and defined in the following. Under a *grammar*, a set of rules is understood which combine a certain amount of words to sentences. All possible sentences which may be built in a grammar constitute a *language*. Each sentence in the language complies to the rules and is therefore called to be *syntactically correct*. Each sentence which can also be correctly interpreted in a certain domain, i.e. is meaningful, is called to be *semantically correct*. Finally, a sentence is also called to be *pragmatically correct*, if the meaning is valid under the given boundary conditions (BC).

The purpose of design languages is based on a functional analogy: in systems engineering, a design (i.e. a *sentence*) is based on a composition of basic building blocks (i.e. the *word(s)* of the vocabulary). But not every design which can be built by a fixed number of components is technically meaningful. Also, it may not necessarily fulfill the customer requirements in an optimal sense. The resulting sets of the syntactic, semantic and pragmatic correct sentences (visualized in Figure 1 as a grey bubbles) are therefore

becoming smaller and smaller. The possibility to compute these sets gives rise to something which may be described as 'computer-based design space exploration' (Rudolph 2002) and may be a key to future conceptual design, if the validation problem of these computer-generated design can be adequately resolved similarly to the mentioned domain of VLSI chip design.

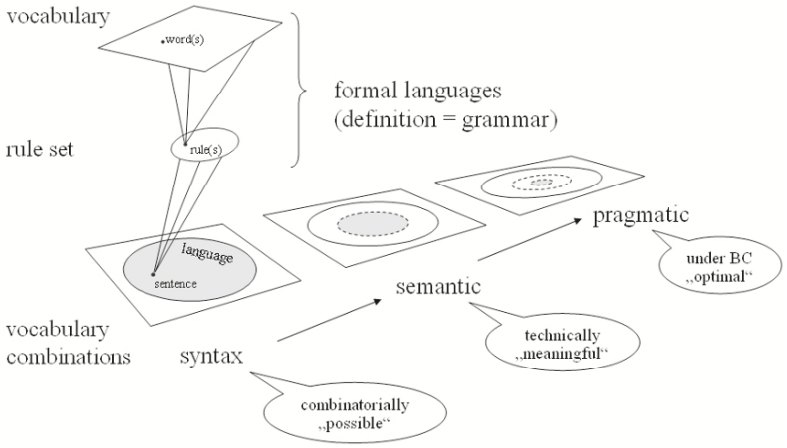


Figure 1. Definitions of syntax, semantic and pragmatic (Rudolph 2003).

Usually, the syntactic correctness of grammar-based design transformations is relatively easy to verify and it would be a standard procedure to extend this approach to the verification of semantics and pragmatics by the establishment of the syntactic correspondences for the desired semantic and pragmatic design aspects. However, in cases where this is impossible to achieve, other means have to be found to guarantee for the semantic correctness of a design rule in an graph-based engineering design grammar.

In the broad area of engineering however design languages based on graph grammars can not (or at least not yet) profit from the projection of all semantic design aspects onto syntactic operations. Consequently, all three aspects of syntactic, semantic and pragmatic correctness of an automatically generated design need therefore to be verified and validated separately. In practical applications of design languages, especially the semantics of the individual design steps are yet by far too complex to be successfully mapped to syntactic operations only. For this reason a methodology to establish a validation scheme for graph-based design grammars is investigated.

The achievement of this goal is attempted through the introduction of the concept of a semantic hull for design object concepts. This means that the rule expansion is semantically verified by the human engineer in a reasoning process about the correctness of the conversion of the semantic hulls in a design rule. Since this might be a highly context-dependent choice, the

addition of the necessary context conditions will make a formerly possibly more context-free design grammar become now context-dependent.

This fosters the interpretation of a *design rule* as a generic *design pattern* intended to match a (very) specific design situation. The overall correctness of the semantic rule expansion in the translation process of a design compiler is then claimed to be maintained, since it should remain unaffected by an instantiation of a sequence of incrementally correct semantic rules.

The semantic validation scheme for engineering design grammars is introduced using the analogy of an induction proof in reverse sense and by reference to computer program validation techniques based on pre- and post-conditions. Several examples from a design language for satellites (Schaefer and Rudolph 2005) are used to demonstrate the validation scheme as well as to illustrate the practical implications of the suggested approach.

2. Validation Techniques

Validation techniques or *proof techniques* (Pohlers 1989; Wolter 2001) for short are typically based on the formal properties of the representation form used in the respective domain. Two important proof techniques, one from mathematics (see section 2.1 *complete induction*) and one from computer science (see section 2.2 *formal program correctness*), are shortly reviewed. Then the attempt to establish a validation scheme for similar purposes in the area of formal graph-based design languages is investigated.

2.1. COMPLETE INDUCTION

The proof technique of *complete induction* (Wolter 2001) is based on the properties of the natural numbers and consists of three steps. According to Figure 2, it is first necessary to show that the so-called *induction axiom* or the begin of the induction *E(0)* is a true statement. Secondly, the *induction step E(m)→E(m+1)*, i.e. the fact that *E(m+1)* is true based on the assumption that *E(m)* is true, needs to be shown for any $m \in \mathbb{N}$.

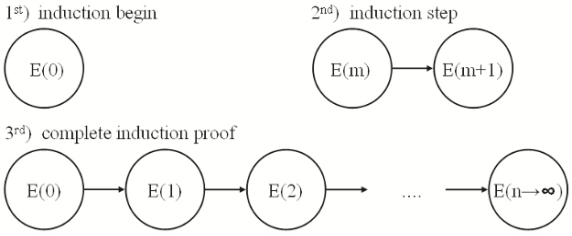


Figure 2. Scheme of induction proof.

Some induction axioms are not $E(0)$, but $E(b)$ for some fixed number $b > 0, b \in \mathbb{N}$. In respect to what is needed later on, it is noteworthy that the

proof scheme works with one single rule only (i.e. the induction step), which is then however applied over and over again.

2.2. PROGRAM CORRECTNESS

Since the software crisis in the 70's, the validation of software programs has become an important issue. Among the most prominent and formal attempts is the *formal proof of program correctness*. It is achieved by the formal representation of the computer program logic, i.e. how the input conditions (e.g. the input data) are transformed line by line of the computer program code into the output condition (e.g. the output data (Schöning 2002).

Mathematically, the transformation of each line of code L_n in n lines of code can be written as a transformation of the pre-conditions $\{V_n\}$ of the n-th line of code to the post-conditions $\{N_n\}$ of the n-th line, usually written as

$$\{V_n\} \quad L_n \quad \{N_n\} \tag{1}$$

Since the post-conditions of the n-th line is the pre-condition of the $(n+1)$-th line, the formal verification works through propagation of the specified input conditions $\{V_1\}$ through all the program code.

$$\begin{array}{lll} \text{(input =)} & \{V_1\} \quad L_1 \quad \{N_1\} & \\ & \{N_1 = V_2\} \quad L_2 \quad \{N_2\} & \\ & \cdots \quad \cdots \quad \cdots & \\ & \{V_{n-1}\} \quad L_{n-1} \quad \{N_{n-1} = V_n\} & \\ & \{V_n\} \quad L_n \quad \{N_n\} & \text{(= output)} \end{array} \tag{2}$$

While the approach to proof program correctness formally by means of the shown notational forms (1) and (2) has quite some mathematical beauty and strength, but it is impractical even for small programs due to the fast growing length of the pre- and post-conditions. Therefore the approach has been abandoned and has been replaced by other means of software testing.

2.3. OPEN ISSUES IN DESIGN GRAMMAR VALIDATION

Both proof schemes in the previous sections are representatives of so-called *formal languages*. These formal languages possess the properties of syntax, semantic and pragmatic. While computers cannot process semantics *per se*, these proof schemes could have been successfully established because they relied on syntax and syntactic properties only. In the case of engineering design languages based on graph-based design grammars, this limitation to syntactic operations however does not hold.

According to Chomsky (2000), the expression power of formal languages forms a hierarchy according to the possible amount of context-sensitivity. It is this context-sensitivity which makes the design procedures so difficult to generalize. Loosely speaking, one could say that the existence of statements such as "*there is always an exception to the rule"* is a strong hint to this experience. However, engineers are usually quite well capable to describe their design procedures in natural language(s). These natural languages are a superset of all formal languages and allow for infinite context-sensitivity. It is therefore hoped that much of the easiness of reasoning and discussing about design can be successfully modeled using a graph-based design language by using the design compiler 43 (IILS 2005) for translation.

However, for a successful industrial application of such a *design by design compilation technology* the important issue of the validation of the automatically generated engineering designs must be resolved.

3. Design Grammars

In order to give a taste of the different existing approaches to design, design languages and grammars, three different design language types along with some designs generated with them are presented here. In order to emphasize the commonalities and the differences to the languages used in mathematics and computer science, the representatives are selected from the group of *string-based*, *shape-based* and *graph-based* design languages.

3.1. STRING-BASED DESIGN GRAMMARS

Lindenmayer introduced the so-called L-System notation (Prusinkiewicz and Lindenmayer 1996) as shown in definition (3). This grammar consists of an axiom ω for the start of the rule expansion and a set of four production rules $P = \{p_1, p_2, p_3, p_4\}$, but even more complex grammars for the modeling of the growth sequence of all kinds of plants and other artificial objects have been conceived (Alber and Rudolph 2002).

$$\begin{aligned} &\omega : \mathrm{A};\\ &p_1 : \mathrm{A} \rightarrow [\&\mathrm{FL!A}]/////[\&\mathrm{FL!A}]///////[\mathrm{FL!A}];\\ &p_2 : \mathrm{F} \rightarrow \mathrm{S}/////\mathrm{F};\\ &p_3 : \mathrm{S} \rightarrow \mathrm{FL};\\ &p_4 : \mathrm{L} \rightarrow [\text{''''}\hat{}\hat{}\{\text{-f+f+f-|-f+f+f}\}]; \end{aligned} \tag{3}$$

Some of the vocabulary words (here only: +, -, [,] and F, but not A, S and L) are later interpreted by a so-called *turtle graphic* (Abelson and diSessa 1984) geometrically. For the choice of a branching angle $\beta = 22.5$ degrees, geometry in Figure 3 is generated (Sabatzus 1996). L-Systems are inherently adapted to model tree-like topologies only, so for the modeling of graph-like topologies in engineering other approaches have been developed.

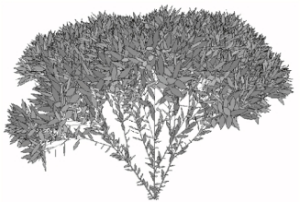


Figure 3. L-system generated plant (Prusinkiewicz and Lindenmayer 1996).

3.2. SHAPE-BASED DESIGN GRAMMARS

Agrawal and Cagan (1997; 1998) conceived a shape grammar for the design of coffeemakers. To give an impression of how the design knowledge is expressed in shape grammar rules, two design rules are shown in Figure 4.

Figure 4. Two shape rules in coffeemaker design (Agrawal and Cagan 1997).

It is important to notice that the rules in Figure 4 are directly expressed in geometrical form, this means that all the reasoning about the design, i.e. the reasoning about shape and function and all the other topics relevant to the design must also be expressed in this way or must at least be added to this form. While this has been shown to be feasible for coffeemakers, it seems to be a limitation for the full generality in the handling of arbitrary design problems. Figure 5 shows two selected designs out of the huge number of designs generated with the shape grammar.

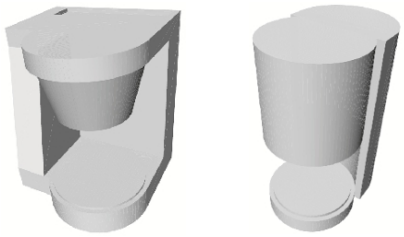


Figure 5. Two coffeemakers from shape grammar (Agrawal and Cagan 1997).

The issue of design validation emphasized there is called ‘maintaining the functionality’ in (Agrawal and Cagan 1997; 1998) but is only implicitly treated, since it is stated that for a design to remain valid the ‘water heater must be below or within the water storage unit’, so that the heating function is always preserved.

While the focus in the previous approach was on the generation of innovative and the reproduction of existing designs, Shea and Cagan (1998) and Shea and Smith (1999) conceived a shape grammar for the generation and shape annealing of existing transmission towers. Their rules are mostly concerned with modification of topological and parametrical properties. Some of the design rules are reproduced in Figure 6 for illustration.

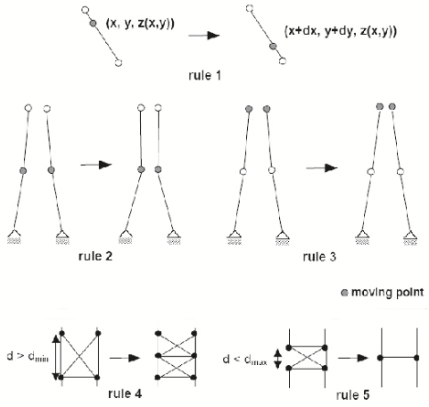


Figure 6. Five rules for transmission tower design (Shea and Smith 1999).

An important part of the optimization is played by the rules for the shape annealing process, since the modification of a highly constraint existing design differs from design rules originally used in the design generation. As an illustration, Figure 7 shows a tower generated by this annealing method.

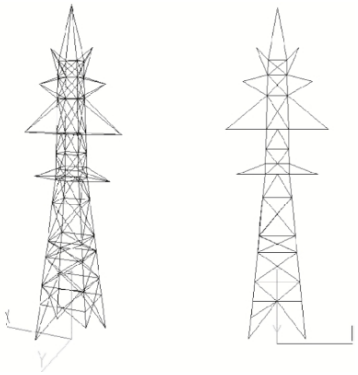


Figure 7. Transmission tower optimized by shape grammar (Shea 1998).

3.3. GRAPH-BASED DESIGN LANGUAGES

As an further step of abstraction in the development of powerful graph-based engineering design languages, a corresponding domain-independent

translation machine called *design compiler 43* (IILS 2005) was implemented in a cooperation of the *Institute for Statics and Dynamics of Aerospace Structures* at the University of Stuttgart with the *IILS Ingenieurgesellschaft für Intelligente Lösungen und Systeme mbH*, Stuttgart (IILS 2005).

Several articles have been published in the context of the design compiler 43 in general (Rudolph 2002; Rudolph 2003), as well as on dedicated applications such as satellite (Schaefer and Rudolph 2005) and car body design (Haq and Rudolph 2004). While these works share the idea of a graph-based design representation with other graph-based approaches, see (Schmidt and Cagan 1996; Schmidt, Shetty et al. 1998) as examples, it represents a quiet different information processing architecture.

The design compiler 43 which is used in all examples in the paper from now on offers an intermediate, domain-independent design representation in form of the so-called *design graph* as it is introduced in the next section. This intermediate representation offers the possibility to incorporate several conceptual advantages in the design information processing architecture (Rudolph 2002). These are (among others):

- *A library concept* which allows the mapping of the graph nodes to a specific domain. Domain dependencies are thus hidden in a library.
- *A constraint processing mechanism* for symbolic equations. The design constraints are thus collected at runtime and solved using a symbolic computer algebra package.
- *A plugin programming technique*, which allows the bi-directional interfacing to any of the typically customer-defined numerical analysis codes (multi-body, finite element, discrete simulation, etc.).

3.3.1. Design Graph

As a specialty of the approach to engineering design underlying the *design compiler 43* (IILS 2005), the so-called *design graph* represents a complete design description at any moment in time. The creation of the design graph starts initially with the axiom and is transformed during the execution of the design rules (i.e. graph transformations, also called *design patterns*).

The axiom consists usually of one single node (such as the node named global node in Figure 8) only, which is expanded to the final design graph during rule execution. However, all the customer requirements are typically added to the axiom, since they are fixed requirements which must be met by any design under development.

Figure 8 shows a complex axiom for a satellite design, where several nodes represent the four required satellite up- and down-links, the need to fly certain orbit maneuvers, the use of a specialized structural adapter to fix the satellite to the rocket and so on. The links between graph nodes signify that certain parameter values in the descriptions of the nodes are coupled and will obtain identical values during the constraint processing stage.

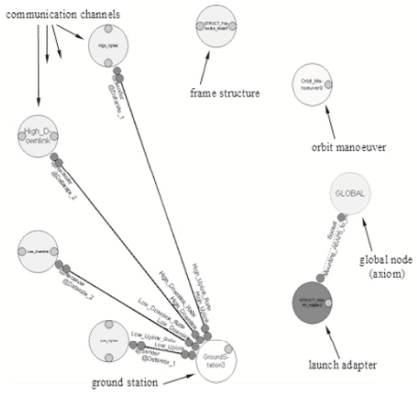


Figure 8. A complex axiom (in satellite design) (Schaefer and Rudolph 2005).

3.3.2. Design Patterns

In Figure 9, the so-called *4-quadrant scheme* (or *modified x-scheme*) is used to define a design rule in form of a graph transformation. Permitting graph transformations allows much more freedom for the designer to express the context of his design decisions than a simple database access mechanism, which tests for the existence of an item in memory only. Additionally, constraints of any kind (logical, symbolical, numerical, etc.) which can be evaluated at runtime may be used throughout the rule execution process.

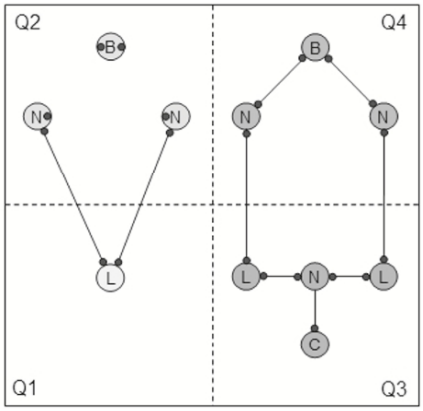


Figure 9. The 4-quadrant scheme for rule definition (Haq and Rudolph 2004).

The four quadrants Q_1, Q_2, Q_3 and Q_4 are used as it is described in the following. The *if*-part of the rule in Figure 9 is specified by Q_1 and Q_2, thus describing the conditions of (sub-)graph matching which must be satisfied so

that the rule can be executed. Hereby all nodes in Q_1 are deleted during the rule execution process, while all the nodes in Q_2 are transferred to Q_4.

The *then*-part of the rule in Figure 9 is specified by Q_3 and Q_4, thus describing the creation of new graph nodes in Q_3 and the modification of the context (i.e. of the connecting lines between the nodes) in both Q_3 and Q_4. While the syntax is completely described by the above rule, the semantic meaning of the nodes *B, N, L* and *C* and their parameter values do depend on the underlying vocabulary definitions.

3.3.3. Design Expansion

Based on the intermediate stages of the design graph during rule execution at time *t* as shown in Figure 10, the design rule in Figure 9 transforms the design graph to its new form in Figure 11 at time instant *t+1*.

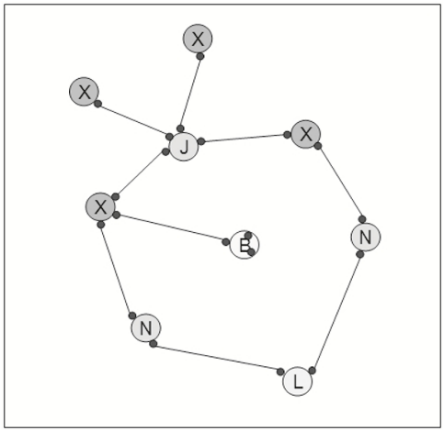


Figure 10. Design graph at time *t* before rule execution (Haq and Rudolph 2004).

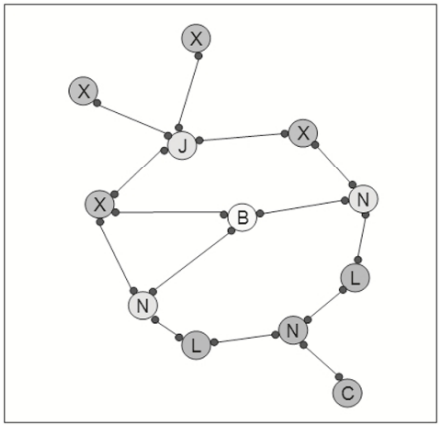


Figure 11. Design graph at time *t+1* after rule execution.

Since the *if*-part in Figure 9 is matched with the (sub-)graph in Figure 10, the rule can be executed. (In fact, the two nodes *N* are both connected to *L*, and the node *B* exists.) Then node *L* in Q_1 is deleted and due to the *then*-part of the rule the nodes *L*, *N*, *L* and *C* are created. Finally, the modifications of the context lines lead to the modified design graph in Figure 11.

The rule scheme is therefore very practical to express design context, since the preconditions can be made as precise and specialized or as general as necessary. Of course, in respect to rule reuse, the most general design rule formulations are desirable (Haq and Rudolph 2004).

3.4. DESIGN PROCESS SEQUENCE MODELING

Since a design grammar consists of a vocabulary, an axiom and a set of rules to describe a certain design sequence, the main two design principles (i.e. a *top-down* versus a *bottom-up* approach) behind it are described first.

3.4.1. Top-Down Sequences

In a top-down design sequence, which is typical for systematic product development processes such as described in the engineering norm VDI 2221 (VDI 1986), the design flow goes *"from the abstract to the concrete"*. First, the given customer requirements (often expressed in the axiom) are mapped onto abstract functional requirements. The functional requirements are then transformed in concrete functions by means of solution principles. To determine the final design, the concrete functions are mapped on the components. The final spatial components outline (the so-called *package*), along with the generation of the necessary connections (the so-called *piping* and *routing*) represent the last steps in this top-down design philosophy.

The most important property of the top-down approach is to go in each design step from a more abstract to a more concrete design description. Each later design stage description therefore represents semantically a subset of the earlier design description. Besides the presence of the so-called *side-effects*, which seem to be unavoidable due to the dimension expansion going along with any concretization is described in Section 4.3.1 in more detail. However, this subset property is later central to the validation scheme.

3.4.2. Bottom-Up Sequences

In a bottom-up design sequence the design process seems to work much less predetermined way. This is however a misconception. A bottom-up design approach is typically adopted when complex physical couplings exist which make an independent modular, thus 'additive way' of designing impossible. A known example is the structural body design for crashworthiness. In all such cases, where global design rules are unknown or difficult to devise a priori, the design space is explored by imposing geometrical changes in the synthesis stage locally which are followed by a numerical analysis stage, where the global field problem is solved by numerical integration.

While bottom-up and top-down design are antagonistic principles, they frequently intermix due to the concurrency of both strongly and weakly coupled field problems in design. However, both can be easily modeled in a rule-based design language. For the bottom-up design approach this has been shown in Section 3.1 for an L-system. For the top-down approach this is shown using the example of a set of rules stemming from a satellite design sequence in the following.

3.4.3. Satellite Design Sequence

A satellite design language (Schaefer and Rudolph 2005) was developed for either interactive or batch mode execution in the design compiler 43 (IILS 2005). The design follows a top-down approach from mission requirements (i.e. the customer requirements) to the final embodiment. The actual overall design sequence consists of more than 120 rules in the described *x*-scheme notation from section 3.3.2 and about 100 potential vocabulary elements with mostly symbolic constraint information about their functionality and physical behavior. More details about the design language can be found in (Schaefer and Rudolph 2005). For the reason of space, only the rules in presented in Figures 12, 13 and 14 will be used in the validation scheme.

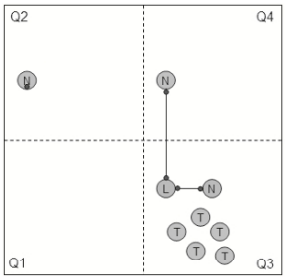


Figure 12. Propulsion rule (Schaefer and Rudolph 2005).

Based on the requirement to fly a certain orbit, the propulsion rule in Figure 12 defines the generation of a chemical rocket motor *L*, the nozzle *N* and five tanks *T*. The rocket motor and the nozzle are already explicitly coupled by a node link, which is not yet the case for the tanks. This is done because the propulsion can be finalized only after the mass balance has been determined, thus leading to an increase or decrease in the number of tanks later on, making the manipulation of the connectivity information (i.e. for the piping) at this point in time unnecessary.

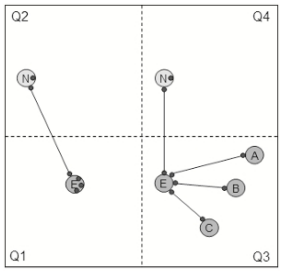


Figure 13. Energy scheme rule (Schaefer and Rudolph 2005).

Figure 13 resolves the overall static energy balance into a predefined number of energy schemes which are resolved in time. These are the three operational modes (e.g. the *safe mode A*, the *nominal mode B* and the *payload mode C*) in the shown energy scheme rule which are used to detail the static energy consumption in the satellite by integrating the different needs over the duty cycle time (e.g. over one or several orbits, depending on the mission requirements). This is beneficial because not every electrical load may be on all the time.

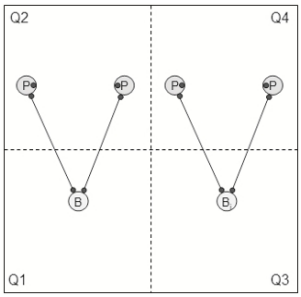


Figure 14. Joint substitution rule with screw (design rule $B \Rightarrow B_i$).

Figure 14 defines the concrete embodiment of an *ideal joint B* in form of a structural connection with a screw B_i. Imaginable alternatives would be structural connections using *rivets* B_j, *welds* B_k, *bonds* B_l, and so on. In this respect the design rule just selects a special solution out of the named options. In mathematical terms, *screws* B_i are therefore a true subset of all joints B, so $B_i \subset B$ holds, since $B = B_i \cup B_j \cup B_k \cup B_l$ is the generic term for all available structural connection forms known to the rule set.

4. Design Grammar Validation

In this section, a validation scheme for graph-based design languages is developed. In a direct comparison to the proof techniques already presented in Sections 2.1 and 2.2, it is worth noting that in comparison to mathematical expressions, which are subject to an induction proof based on syntactic operations only, the graph-based design languages possess all three aspects of formal languages such as syntax, semantic and pragmatic (both in form of the design graph representation and the design rules) to the outmost extent.

Since computers as a matter of principle can treat syntactic operations only, it is essential in order to establish a reliable validation scheme for graph-based engineering design grammars to decide how the missing but most essential aspects of semantic and pragmatic correctness are accounted for. This validation procedure is established in detail in the following.

4.1. SEMANTIC VALIDATION SCHEME

Graph-based design languages possess the formal aspects of syntax, semantic and pragmatic which need to be checked for correctness. In the following validation scheme, these three aspects are treated as follows: The

- *syntactic correctness* is guaranteed by the correctness of the sub-graph matching, since the design graph is created based on the definition of the axiom and the graph transformations of the design rules. Since the connections between graph nodes are made through ports, additional type checks (similar to the ones known from string-based computer languages) may be executed at compile time.
- *semantic correctness* is guaranteed by the human design engineer, who needs to validate both the final design graph as well as all the individual design rules (in this respect, the axiom accounts for the very first design rule) in between. The semantic correctness is then claimed to be unaffected, because the semantics of the design graph is changed by a semantically correct design rule only.
- *pragmatic correctness* is guaranteed by a constraint processing mechanism, which uses a symbolic computer algebra package for all operations. Besides the declaration of the design constraints in form of symbolic equations inside the vocabulary and the numerical values of the boundary conditions in form of design requirements, no other human interaction occurs at compile time.

The above three levels of syntactic, semantic and pragmatic correctness are used in the following to establish a proof scheme in the *reverse sense* as will be explained later around Figure 15. 'Reverse' means here 'in the inverse direction' as the proof scheme in Figure 2, thus starting with a semantically validated final design solution already. The validation aspects of syntax and pragmatic are uncritical, since they can be projected (by means of sub-graph matching and constraint processing algorithms) on the correctness of syntax operations, whereas the semantic correctness cannot be treated by computers *per se* and remains a domain reserved for human reasoning and thought.

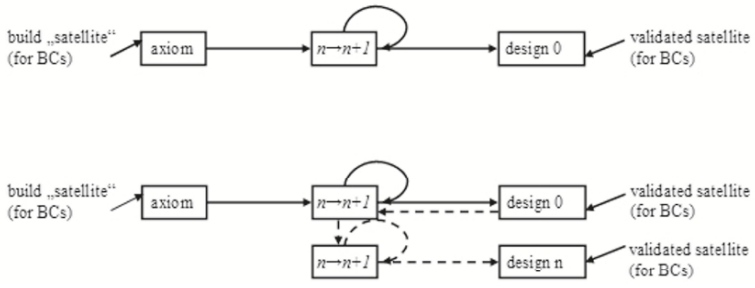


Figure 15. Semantic validation scheme (top-down satellite design).

Figure 15 as a validation scheme for graph-based design grammars works as follows. First, the *human engineer* validates semantically the *axiom*, each of the *design rules* of the graph-based design language and finally one *design solution* which has been generated by a rule expansion of the axiom, i.e. boundary conditions (BCs). As a result, the complete design sequence (e.g. the solid arrows from left to right in Figure 15) is validated.

As a core element in this semantic validation by a human engineer it is stressed that semantic validation means that a human verifies the semantic meaning of the axiom, of each of the graph rules as well as of the finally resulted design. This is done by using the thought construct of a *semantic hull* of a concept. Examples of semantic hulls are visualized in Figure 16.

According to the standard definitions used in philosophy (Ganter and Wille 1996), a concept B has extensional and intensional properties. The *intensional properties* describe all *features* of the concept, while the extensional properties consist of all *objects* summarized under the concept. In the sense of the design rule in Figure 14, which serves as a substitution rule of an abstract structural connection *joint* (denoted mathematically as B) with a concrete realization *screw* (denoted mathematically as B_i) this concretization proved because of $B_i \subset B$ holds.

Using the notation $B_i \subset B$ of set theory as shown in Figure 14, the semantic validation of a design rule can now be verified relatively easy and straightforward by a human: Since the super-concept B is replaced by a sub-concept B_i inside the design rule $B \rightarrow B_i$, it is guaranteed that the substituted semantic hull becomes smaller, thus narrowing (top-)down the design representation from the abstract to the concrete.

The validation scheme in Figure 15 works as follows: Given the correctness of the final design, the correctness is not altered if in the last rule $B \rightarrow B_i$ with the optional rule $B \rightarrow B_j$ another, equally permissible concept concretization $\mathrm{B_j}$ with $\mathrm{B_j} \not\equiv \mathrm{B_i}$ but $B_i \subset B$ and $B_j \subset B$ is chosen, as had been semantically verified by the design engineer in the first hand. This is indicated by the dashed arrows in Figure 15 and means that another object with other features (or the same object with different features) has been chosen in this modified detailing rule. Of course, what can be done with the last rule may done with the second last one and so on, thus leading to a chain argument of semantically correct design modifications.

4.2. SATELLITE EXAMPLE

In the following, the three design patterns (i.e. the rules from Figures 12, 13 and 14) will be discussed again for exemplification and further clarification of the semantic validation scheme for graph-based design grammars which was introduced in the previous section.

4.2.1. Design Pattern for a Structural Connection

Figure 16 illustrates the concept of the semantic hull for the example of the aforementioned replacement rule of an abstract structural joint by a screw connection in Figure 14. The semantic validation of the design rule *(joint → screw)* corresponds directly to the way humans typically express their design experience with words in a natural language. Noteworthy is also that additional constraints may be used to add the necessary design context sensitivity, thus making the design rule a generic, reusable design pattern.

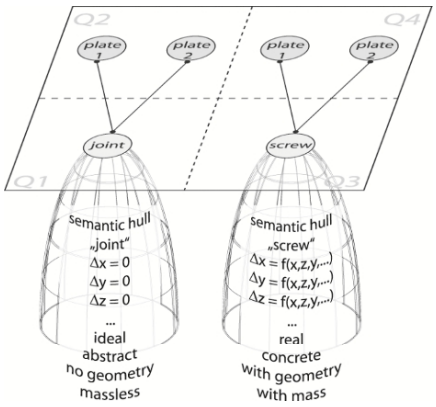


Figure 16. Joint resolution rule with screw.

Due to the concretization, additional *side-effects* may occur, as is discussed again in section 4.3.1. While an ideal structural joint has a priori no mass or geometry, these additional components of the design description are locally created and must afterwards be checked for global consistency. This means that after modifications system balances and collision checks must be re-executed, which can be achieved by a proper placement of the joint rule in the overall sequence of design patterns in the design language.

Finally it is noteworthy that the design patterns add their design knowledge incrementally to the design representation during the creation of the design graph. This means that the above design rule *joint → screw* just takes the (intermediate) decision for further concretization of the concept *joint* by replacement of the concept *screw* at this design stage. However, about the final parameters, i.e. the pragmatics of the particular screw as shown in Figure 17, nothing is determined yet. These screw parameters are determined later on by a more specialized design patterns (VDI 2230 2003).

4.2.2. Orbit and Energy Design Patterns

The rule semantics of Figures 12 and 13 can be validated similarly. Both rules, the propulsion rule *orbit maneuver → rocket motor, nozzle, 5 tanks*

and the energy operations rule *static energy balance* → *safe mode A, nominal mode B, payload mode C* can be semantically validated by an (or a team of) experienced satellite design engineer(s).

A design language may be seen in this respect as the incremental decomposition of the act of designing into machine readable design patterns with semantic hull substitutions. A well chosen vocabulary allows an almost *1:1* correspondence between the wording of the designer(s) and the chosen concept abstractions in the graph language, thus minimizing the extra effort of using an artificial language for design knowledge representation purposes.

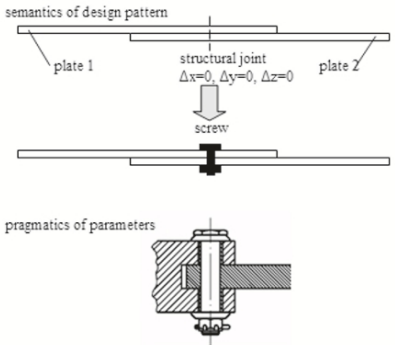


Figure 17. Joint resolution rule with screw.

4.3. DISCUSSION OF ASSOCIATED PROBLEMS

Several issues arise in the downstream of the semantic validation scheme. While some of them may be immediately solved, some of them may remain unsolvable in the next future. Some of these issues are now discussed here.

4.3.1. Side-Effects

Side-effects are typical to any concretization activity in design. Since every object has a mass, volume, consumes/absorbs/emits energy and so on, all of the important system balances need to be (re-)checked after each concretization step. However, despite the fact that this may tend to numerous design iteration cycles, the satellite design appears to be much more well-behaved than expected. Of course, side-effects can theoretically never be excluded and are therefore one of the major sources iteration in the design process.

As a further consequence, the existence of side-effects puts emphasis on the relative position and order of a design pattern in respect to all other design rules. However, this experience just underlines the known context-dependency of the design activities in the course of the design process and does not account for a special drawback of design languages.

Finally to mention, that the problem of side-effects may become more and more complex, as more and more large topological changes may be introduced into the design pattern sequence in Figure 15. Theoretically, a new design object could be created which is claimed to be a satellite by the design compiler 43, but for a human it would not "look like one" at first (e.g. a human would not recognize it as one). However, the new design could be interactively replayed, so the machine could interactively replay the design expansion (i.e. the concretization of the semantic hull of a super-concept with the semantic hull of a sub-concept) to the human by step by step.

5. Summary

A validation scheme for graph-based engineering design grammars has been presented. It is based on the introduction of the concept of semantic hulls of the graph language vocabulary. Based on this, it is shown how a semantic validation scheme for engineering design grammars could be created despite the fact that computers are incapable of processing semantics *per se*.

The semantic validation of a graph-based design grammar has been introduced using an induction proof in the reverse sense and is based on the semantic validation by humans of the correctness of 1) the design axiom, 2) the individual design rules and 3) a final design. The validation scheme however works for strict top-down design sequences only, since bottom-up design sequences represent a continuing abstraction instead of a further concretization step, where the necessary condition of remaining inside a semantic hull does not hold.

The correctness of a top-down design generated with modifications during a rule expansion is claimed to be maintained since it remains unaffected by instantiation of another correct rule(s) inside the semantic hull of the super concept rule(s). Examples from a design language for satellites are used to illustrate the validation scheme derived.

References

Abelson, H and diSessa, A: 1984, *Turtle Geometry*, MIT Press, Cambridge.

Alber, R and Rudolph, S: 2002, On a grammar-based design language that supports automated design generation and creativity, *Proceedings IFIP WG5.2 Workshop on Knowledge Intensive CAD (KIC-5),* Malta, Malta.

Antonsson, E and Cagan, J (eds): 2001*, Formal Engineering Design Synthesis*, Cambridge University Press, Cambridge.

Agrawal, M and Cagan, J: 1997, Shape grammars and their languages – a methodology for product design and representation, *ASME Design Engineering Technical Conferences and Computers in Engineering Conference,* DETC97/DTM-3867.

Agrawal, M and Cagan, J: 1998, A blend of different tastes: the language of coffeemakers, *Environment and Planning B: Planning and Design* **25**(2): 205-226.

Chomsky, N: 2000, *The Architecture of Language*, Oxford University Press, Oxford.

Fensel, D: 2004, Ontologies: *A Silver Bullet for Knowledge Management and Electronic Commerce*, Springer, Berlin.

Ganter, B and Wille, R: 1996, *Formale Begriffsanalyse*, Springer, Berlin. (Also : Ganter, B and Wille, R: 1999, *Formal Concept Analysis*, Springer, New York.)

Haq, M and Rudolph, S: 2004, *"EWS-Car" - Eine Entwurfssprache für den Fahrzeugkonzeptentwurf*. VDI Bericht 1846, Verein Deutscher Ingenieure VDI, Düsseldorf.

Horstmannshoff, J: 2002, *System Synthesis of Complex Building Blocks from Multirate Dataflow Descriptions*, PhD Thesis Technical University of Aachen, Shaker, Aachen.

IILS Ingenieurgesellschaft für Intelligente Lösungen und Systeme mbH: 2005, Design Compiler 43 and Entwurfscompiler 43 are trademarks of IILS mbH. Available Online: http://www.iils.de.

Pohlers, W: 1989, Proof theory. An introduction, *Lecture Notes Mathematics* 1407.

Prusinkiewicz, P and Lindenmayer, A: 1996, *The Algorithmic Beauty of Plants,* Springer.

Rudolph, S: 2002, *Übertragung von Ähnlichkeitsbegriffen. Habilitationsschrift, Fakultät Luft-u nd Raumfahrttechnik und Geodäsie*, Universität Stuttgart.

Rudolph, S: 2003, Aufbau und Einsatz von Entwurfssprachen für den wissensbasierten Ingenieurentwurf. 3. Forum Knowledge-Based Engineering, Stuttgart.

Sabatzus, P, 1996: Available Online: www.math.tuwien.ac.at/~sleska/html/w95html/mathlb95/mathlb95.htm

Schaefer, J and Rudolph, S: 2005, Satellite design by design grammars, *Aerospace, Science and Technology (AST)* **9**(1): 81-91.

Schmidt, L and Cagan, J: 1996, Grammars for machine design, in JS Gero and F Sudweeks (eds), *Artificial Intelligence in Design*, Kluwer Academic Press, pp. 325-344.

Schmidt, L, Shetty, H and Chase, S: 1998, A graph grammar approach for structure synthesis of mechanisms, *Proceedings 1998 ASME Design Engineering Technical Conferences, DETC98/DTM-5668.*

Schöning, U: 2002, *Ideen der Informatik*, Oldenbourg, München.

Shea, K and Smith, I: 1999, Applying shape annealing to full-scale transmission tower redesign, *Proceedings of the 1999 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, Las Vegas, NV, DETC99/DAC-8681.

Shea, K and Cagan, J: 1998, Topology design of truss structures by shape annealing, *Proceedings of the 1998 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, DETC98/DAC-5624*, Atlanta, GA.

Stiny, G: 1980, An introduction to shape and shape grammars, *Environment and Planning B: Planning and Design* **4**: 89-98.

Synopsis Inc: 2004, *Design Compiler*, Available Online: http://www.synopsis.com.

VDI-Richtlinie 2221: 1986, *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte*. VDI-Verlag, Düsseldorf.

VDI-Richtlinie 2230: 2003, *Systematische Berechnung hochbeanspruchter Schraubenver bin-dungen*. Blatt 1, VDI-Verlag, Düsseldorf.

Wolter, H: 2001, *Lexikon der Mathematik*, Band 1 (A bis Eif), Eintrag Beweismethoden: (4. Induktionsbeweis). Spektrum Akademischer Verlag, Heidelberg.

LEARNING IN DESIGN

Inductive machine learning of microstructures
Sean Hanna and Siavash Haroun Mahdavi

Learning from 'superstar' designers
Paul Rodgers

The improvement of design solutions by means of a question-answering technique
Constance Winkelmann and Winfried Hacker

Contextual cueing and verbal stimuli in design idea generation
Lassi Liikkanen and Matti Perttula

INDUCTIVE MACHINE LEARNING IN MICROSTRUCTURES

Estimating a finite element optimisation using support vector machines

SEAN HANNA AND SIAVASH HAROUN MAHDAVI
University College London, UK

Abstract. A support vector machine is trained to produce optimal structures. The problem of structural optimisation is typically solved by a search procedure that samples and repeatedly evaluates a physics-based model, but this process is computationally demanding. Instead, the use of a learning algorithm to generate new structures based on previously optimized examples is described that provides enormous computational saving. The results have shown that the predicted structures are accurate, and the process is highly efficient for cases in which similar optimisations must be performed repeatedly, especially as the number of such optimisations grows.

1. Introduction

Nature builds by trial and error, via the very effective but slow and costly process of evolution. As humans, our capacity to learn from experience has given us the ability to engineer and build based on our knowledge, and while our designs may not outdo nature in all her complexity, they can excel when the problem is simple and well defined. An engineer can design a building or bridge that is structurally sound without the need for failed attempts. Although for several centuries the mathematical tools for explicit analysis have been dominant, the vast majority of design decisions throughout history have been based on experience of precedents: a practiced builder would know what would stand or fall without having to test it. In a similar fashion, this paper demonstrates that nearly optimal solutions to a well defined structural design problem can be found by training a machine learning algorithm on examples of other solutions found by a traditional optimisation procedure.

Once trained, the advantage of such a machine is the same advantage that the human builder's training and experience give: the ability to build quickly and without failed attempts. A structural problem is chosen that involves the repeated optimisation of many interconnected modules, and thus takes full

J.S. Gero (ed.), Design Computing and Cognition '06, 563–582.
© 2006 *Springer. Printed in the Netherlands.*

advantage of this increase in speed. It is also a problem of sufficient complexity that the solution can not be calculated directly, but must be found by slow simulation and testing. An algorithm capable of arriving at a general solution by inductive learning on presented examples is thus highly beneficial.

Given a parameterized structure and a set of loading conditions, it has been shown that various optimisation algorithms can be used to design an effective shape to counter the given load. Search procedures including gradient descent (GD) and genetic algorithms (GA) make repeated evaluations of the strength of different structures to do this (Schoenhauer 1996; Von Buelow 2002; Chen 2002; Hanna and Mahdavi 2004). If the load conditions change, the optimal structure will also be different and the optimisation can be rerun to find the new shape.

This process is time consuming however, requiring repeated iteration for each new design, and is subject to error due to local optima in the search space. This paper uses inductive learning to eliminate the need for this iterative step once sufficient examples have been generated to save processing time and achieve more constant fitness of solutions. If the optimisation is repeated many times for many sets of loading conditions, the optimal shape of the structure can be considered a function of the load. The work in this paper uses a support vector machine to learn this function of optimal structures given the tensile or compressive loads in each axis, and results in a very efficient and accurate alternative to iterative optimisation.

2. Background

The work presented here draws on previous structural optimisation research by the authors, but extends this by replacing the optimisation step with learning. Before addressing the problem, this section provides a background of related research. First, the particular structural problem is defined, followed by a review of relevant structural optimisation and machine learning methods.

2.1. BACKGROUND: THE STRUCTURE

Space frame structures are investigated in this work: a set of linear members oriented in any direction in 3-dimensional space, and connected at node points either by rigid or flexible connections. The specific problem addressed is that of small scale microstructures, an example of which is shown in the photograph, Figure 1. The overall dimensions of this object as fabricated are 1cm ⊠ 1cm ⊠ 2cm, and the individual struts within it are less than 1mm in length. Relevant aspects of the design method will be briefly reviewed in this section.

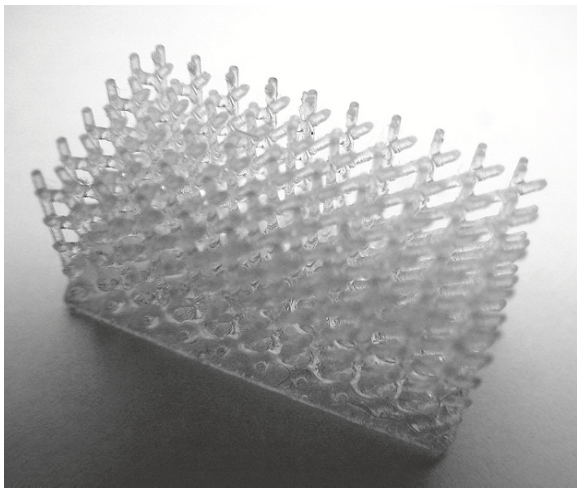
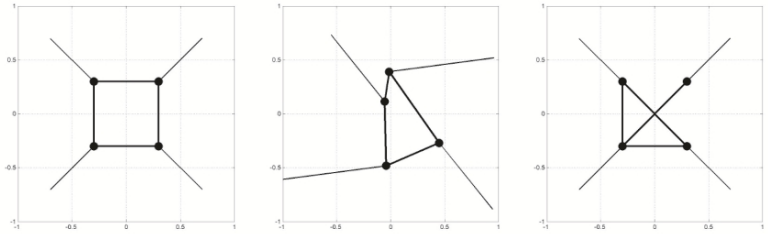


Figure 1. A modular structure fabricated by stereolithography.

2.1.1. Defining the structures

To define a particular space frame one must specify both the members themselves, and the locations and orientations of the nodes in 3-dimensional space. We refer to these as the topology and geometry of the structure respectively.

The distinction between geometry and topology can be described by an example 2-dimensional illustration. Geometry refers specifically to the positions in space of the node points joining the structural members. The following diagrams are of two structures with the same topology but different geometries. As can be observed, the connections and number of members are the same, but the coordinates and orientations of these members differ, Figures 2(a) and 2(b). Topology refers to the structural connections between the node points. A change in the topology of a structure is a change in the number, or way in which the members are connected, Figures 2(a) and 2(c).



(a) (b) (c)

Figure 2. An illustration of a change in the geometry and topology of a structure.

2.1.2. Intended structure and fabrication

The structural problem considered in this work is one based on a modular topology, so that it can be scaled to volumes of any size. A large volume of structure can be subdivided into a grid of cubes, to which we refer as 'unit cubes', each containing a portion of structure with identical topology such that each is connected to its neighbours to form a continuous structure, Figure 3.

Previous work by the authors has resulted in a method for optimising large and complex structures very efficiently using this modular 'unit cube' approach. (Hanna and Haroun Mahdavi, 2004) An object under a complex loading condition exhibits differing stresses at various points in its volume. If these stresses are sampled at the location of one of the unit cubes, they can be used to optimize the module of structure within that cube. The vector of stresses in the three (x, y and z) axes represents a loading condition for the structure in that cube, and for each stress vector there is an optimal set of node point positions and strut thicknesses to best resist that load. Both genetic algorithms and gradient descent (Haroun Mahdavi and Hanna, 2004) have been used to find this optimal, using the finite element method to simulate the effects of loading.

The ideal result is a modular structure as displayed in Figure 3 (bottom), with gradual changes in the geometry of the structure as the stresses change continuously across the volume of the object. It is very efficient, with material concentrated in high stress zones and internal struts aligned to counter the changing direction of the stress vectors. To arrive at this, the optimisation of structural units must be made repeatedly, once for each unit cube of differing stress. The structural problem is therefore similar to time series problems of control and dynamic systems, but static: instead of changing in time, the geometry morphs in space. Because a similar type of optimisation must be performed many times, this paper proposes the method of learning the function of optimal structures from a training set of previously optimized geometries.

2.2. MANUFACTURING

The structures considered are designed to be fabricated by a digitally controlled process, the main advantage of which is the low cost of complexity. Such techniques are increasingly used in such large scale manufacturing as automobiles and architecture (Sischka et al. 2004), but the development of smaller scale rapid prototyping technology allows manufacture at scales less than a millimetre.

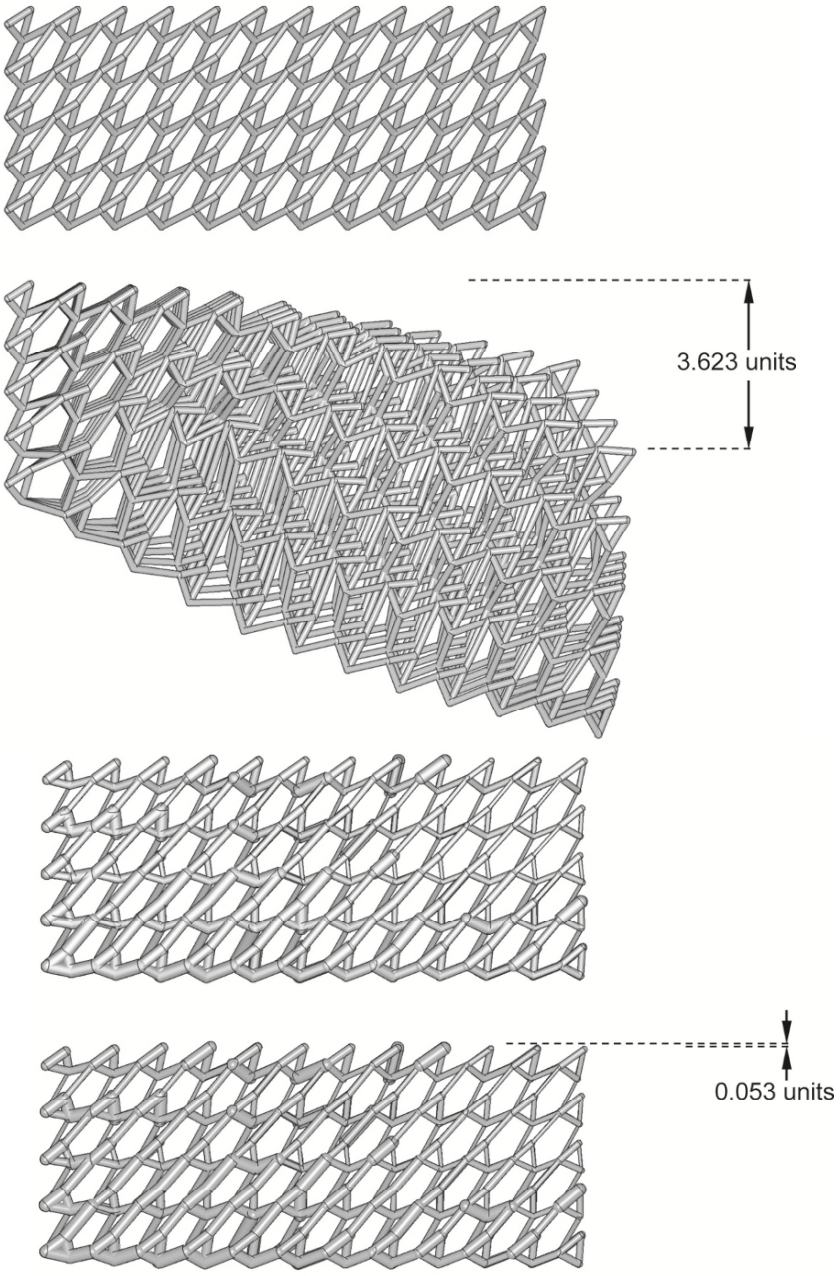


Figure 3. A modular space frame forming a cantilever beam. Both have the same overall mass and topology, but identical modules (*top*) deflect far more under loading than do the individually optimised ones (*bottom*).

Rapid prototyping techniques are now beginning to be investigated as an alternative method of construction for objects of high complexity, particularly with intricate internal structures. This has not yet become commercially viable for mass production, but several researchers are preparing for the increasing accuracy and decreasing cost of the technology in the future. Molecular Geodesisics, Inc. (1999), for example, is investigating structures based on a regular tensegrity space frame which would, at a microscopic size, be useful as biological or industrial filters.

Stereolithography, specifically, is the method considered here. This begins with a tank of liquid photopolymer which is sensitive to ultraviolet light. An ultraviolet laser 'paints' the object as a series of horizontal layers, exposing the liquid in the tank and hardening it. Once completed, the object is rinsed with a solvent and then baked in an ultraviolet oven that thoroughly cures the result. The machines used by the authors are capable of creating very fine structures, and build to a resolution of 0.05 mm.

The horizontal stratification inherent in the process adds a degree of complexity to the problem of optimisation, as members built at different angles to this horizontal plane have varying strengths (Haroun Mahdavi and Hanna 2004). These were measured (Haroun Mahdavi and Hanna 2003) and factored in to the examples presented to the machine for learning.

2.3. OPTIMISATION OF STRUCTURES

Initial data to be used in training any learning algorithm can typically come from several sources, including experts, previously published historical or experimental data, and simulation (Reich 1997). Because of the repetitive nature of the problem and the well defined behaviour of structures, simulation by the Finite Element Method (FEM) is both the most efficient and accurate. In the design task under consideration here it is a set of optimal solutions that is required.

Several techniques have been devised for generating the topology of continuous solids analysed by FEM. Both GA and non-random iterative methods have been used. Marc Schoenhauer (1996) reviews a number of GA methods for generating topology in 2D or 3D space to optimise structural problems involving continuous shapes, in which the genetic representation can determine a configuration of holes and solid using Voronoï diagrams or a list of hole shapes. Yu-Ming Chen (2002) uses a non-random iterative process of shifting node points in the FEM representation toward high stress zones to examine similar problems. These methods can determine the number and position of holes in a cantilevered plate, for instance, but do not deal with truss-like structures.

Discrete element structures (e.g. trusses, space-frames) of the kind considered here involve both the design of the topology of connections, as

well as their position and size. Much early research in this area has been in refining only the shape or member sizes, rather than the topology (in terms of members connecting the node points of the structure). Adeli and Cheng (1993) use a GA to optimise the weight of space trusses by determining the width of each member in a given structure. The shape and load points are fixed in advanced, and the cross sectional areas of groups of members are encoded in the genome, then selected to minimize the total weight.

More recent research has concentrated on topological optimisation, or both topology and shape together. Steel frame bracing topologies for tall buildings have been designed by GA, either by encoding the possible member connections within each structural bay in the genome (Kicinger et al. 2005, Murawski et al. 2000), or evolving a set of generative design rules (Kicinger et al. 2005). Yang Jia Ping (1996) has developed a GA that determines both shape and topology, which must begin with an acceptable unoptimised solution and refine the topology by removing connections. Peter von Buelow (2002) used a two stage algorithm nesting one GA within another. An outer GA evolved a topology for the structure expressed as a matrix representing the structural connections, while another GA found the geometry for each member of the population, expressed as real valued node positions. Previous work by the authors has also used GA for both topology (Haroun Mahdavi and Hanna 2003) and geometry, but it has been found that gradient descent is more efficient for shape optimisation (Haroun Mahdavi and Hanna 2004).

2.3.1. Optimisation by gradient descent

The optimisation performed is gradient descent to minimise the total deflection in a structure under the specified load, as applied to a unit cube. Simulation of this deflection is performed using the finite element method.

2.4. MACHINE LEARNING FOR BEHAVIOUR AND STRUCTURE

Machine learning has long been applied to structures and in the domain of civil engineering, most commonly as an enhancement of the optimisation process. A recurring bottleneck in optimisation is the simulation of a design's behaviour, which can either be time consuming due to the complexity of the model, or simply incorrect due to incomplete knowledge. This can be addressed by 'shallow modelling' a system's observed behaviour with inductive learning (Arciszewski and Ziarko 1990). Discrete, symbolic learning methods have been used to construct rule-based systems, which draw relationships between design parameters that predict the performance of systems from individual beams (Arciszewski and Ziarko 1990) to the steel skeletons of entire buildings (Szczepanik et al. 1996). Sub-symbolic inductive methods such as artificial neural networks have been

used also to predict structural and material performance (Reich and Barai 1999) and the behaviour of mechanical systems such as propeller blades (Reich and Barai 1999; Neocleous and Schizas 1995).

Some of the most recent and complex problems involve structural prediction in the field of bioinformatics, in which the molecular composition of proteins can too computationally expensive to simulate fully. One stream of research is in the prediction of the secondary and tertiary structure of proteins by machine learning, where the inputs are the actual DNA string and the outputs are the predicted three-dimensional structure of the protein. Various learning algorithms have been used, including artificial neural networks (Meiler and Baker 2003) and support vector machines (Wang et al. 2004).

Various machine learning algorithms have been used to find a function to predict movement in time of a dynamic system, which is in some ways similar to structural problems. In both cases the simulation of a physics-based model is possible to an arbitrarily high degree of accuracy, but computationally demanding, and the emulation of this behaviour by a trained learning algorithm is more efficient. The NeuroAnimator uses a neural network trained on physics-based models to produce realistic animation of systems ranging from a pendulum to the swimming of a dolphin. (Grzeszczuk et al. 1998) The method also serves as a control mechanism given a goal (such as balancing the pendulum or swimming toward a target) in the environment, and in this case is similar to the problem of optimisation.

Regardless of the method used in simulation, the repeated iteration of generating and evaluating solutions is the other major hurdle in optimisation. Inductive learning has been found useful to improve the speed and quality of this loop by reusing knowledge of previous designs or iterations. Murdoch and Ball (1996) have used a Kohonen feature map to cluster bridge designs in an evaluation space, and Schwabacher et al. (1998) have used a symbolic learning algorithm, C4.5 (Quinlan 1993), to select appropriate starting prototypes and search space formulations for a parametric optimisation of yacht hull and aircraft designs. Both allow a rapid re-evaluation of previous work which improves the optimisation when run again to new specifications or fitness criteria.

It is the aim of the present work to use a learning algorithm to replace the optimisation process entirely – both the simulation and evaluation loops. While much provious research has concentrated on inferring rules to guide a design (Arciszewski and Ziarko 1990; Szczepanik et al. 1996; Reich and Barai 1999; Neocleous and Schizas 1995), or on suggesting a starting point on which to improve (Murdoch and Ball 1996; Schwabacher et al. 1998), we use induction to derive a function that directly maps a given load condition to an optimal solution.

2.4.1. Algorithm selection

The choice of algorithm is dependent on the learning problem at hand, including the form and availability of data, and the goal of learning (Reich 1997). The goal, in this case, is induction: to derive a generalisation based on previous evidence of optimal structural solutions. Duffy (1997) lists six major machine learning techniques, of which three potentially apply.

- *Analogical* or case-based reasoning techniques explicitly represent past examples in such a way that they can be retrieved and adapted to suit new problems.
- What Duffy terms *induction* – specifically symbolic induction – allows a general rule or pattern to be generated to fit the data. Symbolic algorithms with discrete output such as rough sets (Arciszewski and Ziarko 1990) and C4.5 (Quinlan 1993) above, yield explicit classification or parameter ranges, and have therefore been used to estimate behaviour or recommend design decisions in symbolic or labelled form.
- *Artificial neural networks* are part of a class of sub-symbolic algorithms (including, more recently, support vector machines) that can result in a continuous output, and therefore interpolate exact output values to a finer degree than is specified by the input set. These also perform induction in the form of a continuous function.

The data form is most suited to the third category. The solution to structural shape is naturally a continuous function, and it has been noted that discretisation is detrimental to optimisation performance (in tests by the authors), or can lead to large learning error rates (Reich 1997). As the problem is real valued overall and output is of higher dimensionality than input, it is this sub-symbolic class of algorithms that is appropriate.

3. Learning Methodology

3.1. THE ALGORITHM

Support vector machines (SVM) (Vapnik 1995) are chosen to perform the learning described in this paper. They can be described generally as a type of linear classifier that uses a non-linear kernel function to map input data to a sufficiently high dimension such that it can be separated by a hyperplane (Duda et al. 2001). The transform resulting from this kernel function ensures this hyperplane is non-linear in the original input space, and so the SVM can just as easily be used in regression to a non-linear function as in classification. They will be used in this capacity to learn the function of optimal structures.

Given a data set D, consisting of an input vector $\mathbf{x}$ and a response vector $\mathbf{y}$, the function to be learned

$$\mathbf{y} = f(\mathbf{x}) \tag{1}$$

is approximated by the SVM by building a model $f'(\mathbf{x})$ based on D, that enables the estimation

$$\mathbf{y'} = f'(\mathbf{x}). \tag{2}$$

The type of SVM used by the authors is a Least Squared SVM, in which the solution follows from solving a set of linear equations, instead of quadratic programming for classical SVMs (Suykens et al. 2002). The kernel is the commonly used Gaussian radial basis function.

3.1.1. Learning objective

The design objective to be learned is to find the best structural geometry for a single modular unit given the input of its external load. The task is simply this: for each set of loads, find the set of node points that represent the optimal structure, Figure 4. As a function (1), the input **x** is the three-dimensional vector of external forces corresponding to the stress (in either tension or compression) in the three axes of a given unit cube. This is represented by the components in the directions of the x, y and z axes:

$$\mathbf{x} = (x_{(x)}, x_{(y)}, x_{(z)}). \tag{3}$$

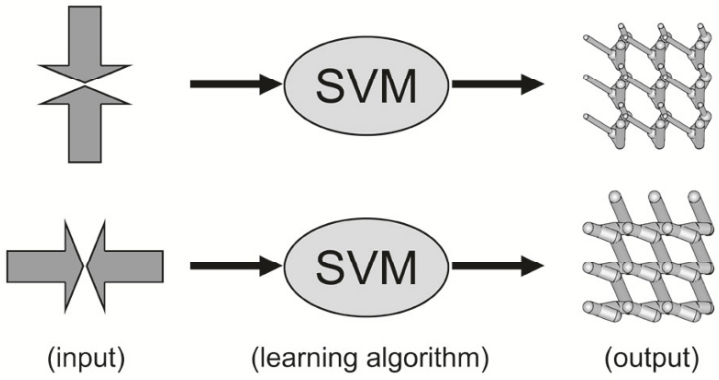


Figure 4. Different force inputs result in ideal geometry outputs.

The output structure **y** consists of the node point positions for the optimal structure as found by prior optimisation: gradient descent as described in Section 2.3. This is the set of (x, y, z) coordinates for each of the node points y_i:

$$\mathbf{y} = (y_{1(x)}, y_{1(y)}, y_{1(z)}, y_{2(x)}, y_{2(y)}, y_{2(z)}, \ldots, y_{n(x)}, y_{n(y)}, y_{n(z)}), \tag{4}$$

The nodes are also located in three-dimensional space, so for a topology of n points the output **y** is a $3n$-dimensional vector.

3.2. THE DATA SET

A single topology was used consisting of four node points per unit cube, resulting in a 12-dimensional output **y**. The data set *D* was created not to uniformly sample the entire space of possible solutions, but for a normally distributed range of forces and the associated optimal solutions.

Each training sample is created by generating a random input vector **x** from the normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$, resulting in a range of approximately [-3:3] units of force in each of the three axes. The actual distribution of each of the components of **x** are plotted in Figure 5. The node point outputs **y** are found by the gradient descent method described in Section 2.3, and result in asymmetrical distributions of node positions throughout the space of the unit cube. The distributions of each of the four node points in the three axes of space are shown in Figure 6. Although the positions of nodes are not constrained by the optimisation algorithm, the repeated nature of the structural modules implies a maximum bound on the search space of one unit for each of the components of **y**. The variance in the data set for each of the points is 0.72, 0.49, 0.53 and 0.56 units in this space respectively, indicating a large portion of the space was sampled in *D*.

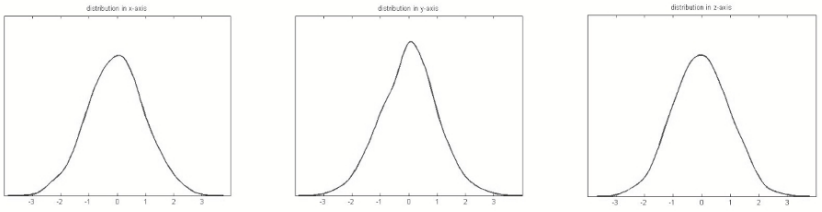


Figure 5. Probability distributions of the x-axis, y-axis and z-axis components of input force vector **x** are based on a normal distribution with mean zero.

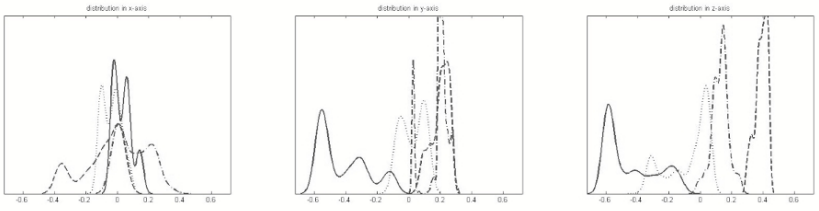


Figure 6. Probability distributions of the x-axis, y-axis and z-axis components of the node points **y** are asymmetrical in physical space, the result of optimisation.

3.3. TRAINING

This work investigates whether the SVM can be trained to predict the optimal geometries of a structure **y** given different force conditions **x**. Each training example is an optimal solution found by the iterated gradient descent algorithm in which each sample is a finite element analysis of the structure. The greatest computational cost is therefore in generating this training data, and so the proposed learning method is online, with a gradually increasing data set rather than as a batch process.

Training of the SVM was performed with the gradually increasing set of stress vectors **x** and node points **y** until the accuracy of the learned function no longer increased. A radial basis function kernel with variance $\sigma^2 = 0.2$. was used to map all dimensions in the SVM.

3.3.1. Error estimation

Methods of error estimation have been systematically evaluated in (Reich and Barai 1999). We have used the most common method, hold-out, which is also the most conservative, in that it maintains a pessimistic bias toward the results.

The data D is divided at random into two sets: a training set T and a separate validation set V. The SVM is trained on T, and then evaluated on V, the errors in V indicating the generalisation error. For D of size n, the size of T is ideally $0.6n$ to $0.8n$ and V is the remaining $0.2n$ to $0.4n$. While there are no general bounds for regression, the data D of size $n > 1000$ produces results with confidence more than 0.95 in classification problems (Reich and Barai 1999).

Our tests conform to these recommendations for accuracy. The performance of the SVM was evaluated for training sets of varying size, to a maximum size $n = 1300$. For all tests, the validation set V was the same, randomly selected set of size 300. The size of D for which the SVM will be considered in our tests to be fully trained occurs at $n > 950$, which is approximately equal to the recommended size for 0.95 confidence, and errors for even smaller training sets have the most pessimistic bias of any estimation method. Our results therefore display the worst-case estimation of errors, and the true accuracy of the algorithm is likely to be no less than is reported in the following sections.

4. The Trained Algorithm: Results and Analysis

To evaluate the results of learning, a SVM was trained on an increasing set T of samples (from 1 to 1000) while being tested against a separate validation set V of 300 samples. In the three graphs below, this performance is evaluated both in terms of how similar the solutions given by the SVM are to the ideal solutions on which it was trained, and how well those solutions

actually perform when tested under given loads. Under both criteria learning was seen to improve steadily with an increasing training set until slightly less than 650 samples were given, at which point the performance plateaued at a very high level.

4.1. ACCURACY OF THE LEARNED FUNCTION

The performance, or error θ, of the algorithm trained with output **y** consisting of a single component y is often measured as a square loss function

$$\theta = 1/n \left[\sum_{i=1:n} (y_i - f'(\mathbf{x}_i))^2 \right] \quad (5)$$

where n is the number of samples in the validation set V (Reich and Barai 1999). As our output vector **y** is 12-dimensional, we generalise this to

$$\theta = 1/n \left[\sum_{i=1:n} \left(\sum_{j=1:d} | y_{ij} - f'(\mathbf{x}_i) |^k \right)^{1/k} \right] \quad (6)$$

where d is the dimensionality of the output vector **y** and k is the exponent of the metric. The choice of k=2 (Euclidian distance) is appropriate for measurement of error in physical space, or k=1 (the Manhattan or city block metric) is suited to independent parameters. As the data in **y** is a combination of both – independent points in physical 3-space – the Manhattan metric of k=1 has been used.

This error θ then is simply the mean distance between the nodes in each of the ideal samples **y** and the nodes in the corresponding solution output by the SVM **y'**= f'(**x**). Distance here is measured by the Manhattan metric (or sum of the difference in each dimension of the 12-dimensional output vectors). The graph below displays the accuracy of the predicted nodes during training with an increasing set T of examples and a separate validation set V of 300 examples, Figure 7. It indicates a steadily decreasing error for training sets T up to approximately 650 (indicated by '○'), at which point there is little further perceptible change.

The number of 650 training examples appears to be a result of the particular data set, rather than inherent in the algorithm, and it is likely the required size of training set T would fluctuate for different structural topologies. There is negligible variance in the resulting error θ when a different randomly selected set T is used in the SVM, or in the order of samples presented in training. While the observed plateau beginning at T size 650 does not coincide with an error θ of zero, it should be noted that both the generalisation of the model f'(x) and the pessimistic bias of hold-out estimation will ensure a lower limit on the error. Training set size 650 is likely simply to be the limit of learning for this problem.

The average accuracy of the function at this point is within 0.005 units to the validation set, or 1/5 the manufacturing tolerance for a unit cube of 2mm.

At this stage the function of optimal geometries as provided by gradient descent can be considered, for all practical purposes, sufficiently learned.

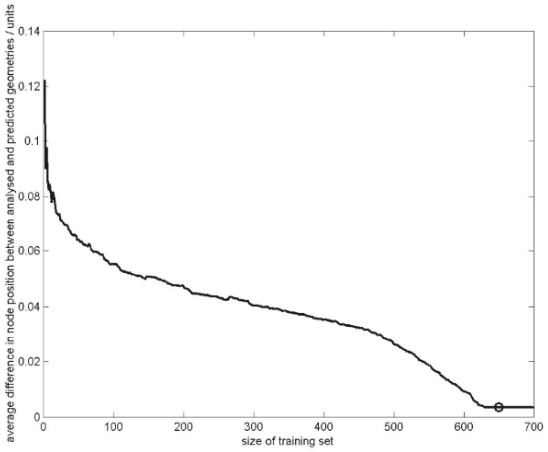


Figure 7. Accuracy of learning increases with increased training.

4.2. PERFORMANCE OF THE PREDICTED GEOMETRIES

While the above graph indicates the standard method of evaluating the accuracy of the function in terms of node distances, it is more relevant to our purposes to know how well the structures *perform* under their respective stresses. This can be determined for a given structure in the validation set by performing a finite element analysis on both the geometry **y** found by GD and the predicted geometry $\mathbf{y'} = f'(\mathbf{x})$ as found by the SVM. Both are loaded with the same input vector of stresses, and their strengths under this load condition are measured as a total displacement of nodes when the load is applied.

This displacement between the original node points **y** and the resulting positions δ under simulated load is thus given by

$$disp(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1:m} [(y_{i(x)} - \hat{y}_{i(x)})^2 + (y_{i(y)} - \hat{y}_{i(y)})^2 + (y_{i(z)} - \hat{y}_{i(z)})^2)]^{1/2} \quad (7)$$

where m is the number of node points, and the performance of the predicted structures **y'** is estimated as the average ratio of displacements

$$\delta = 1/n \left[\sum_{i=1:n} (disp(\mathbf{y}, \hat{\mathbf{y}}) / disp(\mathbf{y'}, \hat{\mathbf{y}}')) \right] \quad (8)$$

where n is the number of samples in the validation set V.

Figure 8 plots this performance δ of the predicted structures **y'** against the same validation set as in Figure 7. A ratio of 1.0 would indicate the predicted structures perform (on average) as well as those found by GD. Again the improvement with an increasing training set is evident over the same range, with a ratio nearly approaching 1.0. The percentage difference between the

resulting displacement of the original samples and the predicted geometries at a training set size of 650 had dropped to 1.51%. Again this occurred slightly at slightly less than 650 samples.

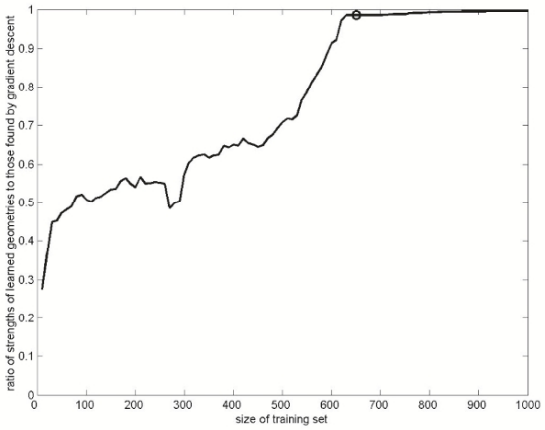


Figure 8. Performance of the structure increases with increased training.

4.3. LEARNED IMPROVEMENTS OVER THE TRAINING SET

As the average performance of structures over the entire prediction set approaches that of the validation set, it can be seen that some predicted structures actually perform better than their equivalents as found by GD. Thus, while the learned function may not be accurate enough to predict the exact node positions in the validation set, in these cases this actually is an advantage, providing an even stronger, more optimal structure. Figure 9 indicates the number of structures of greater strength found by learning for increasing training set sizes. Where 50% would represent the maximum expected value of a perfectly learned function on data with noise, we approach this value at the training set size of 650 with 42% of structures having greater strength than the supposed ideal set.

The fact that many, or indeed any, structures can outperform the optimal structures in the data from which the SVM was trained can be explained by the method in which the data was generated. Gradient descent as a search method is itself prone to error due to local optima in the fitness landscape, and is thus not guaranteed to find the globally optimal solution. Although it has been shown to be an appropriate method for solving the structural shape optimisation problem, it can only do so within an acceptable variance in node positions (Haroun Mahdavi and Hanna 2004). It is this variance that causes some of the optimized geometries in the training and validation sets T and V to fall slightly below the true optimal solution. It can be considered

equivalent to noise in a set of examples collected from real-world measurements. In avoiding overfitting, the regression process performed by the SVM effectively 'smoothes out' the function learned so that some of these optimized structures lie either side of the function $f'(\mathbf{x})$, Figure 9.

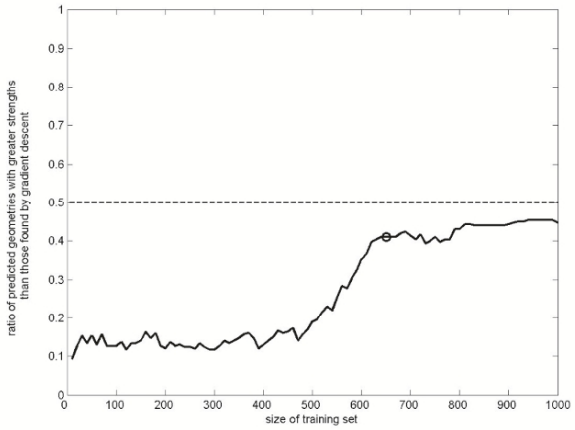


Figure 9. Learned improvements over the training set.

In addition to the ability of the learned function to outperform some of the structures optimized by GD, there is a secondary benefit offered by this smoothing that effects a composite structure formed of many unit cubes. The ideal situation for a complex arrayed structure (as described in Section 2.1) is that stress conditions change gradually and continuously over its volume, and adjacent unit cubes under similar stresses will have similar shaped structure. With any optimisation process applied to individual unit cubes the variance in accuracy, or noise, will cause changes in node position or strut width to be more abrupt between some adjacent cubes. The repeated optimisation of many separate structures amplifies the discretisation caused by the initial sampling of the unit stresses, and these abrupt transitions result in weak points in the overall structure. By using the learned, continuous function to derive the structural geometry, the transitions between adjacent cubes are smoother, and the composite structure benefits in strength.

5. Conclusions

The aim of this work is principally to investigate whether machine learning algorithms, in particular SVMs, could accurately predict the optimal geometries of structures, and thus be used as a substitute for a traditional optimisation algorithm. An SVM was trained on example structures that had been optimized for strength using gradient descent, and used to predict

structures that performed almost as well as an independent validation set optimized by GD. Several conclusions can be drawn from the observations:

- *The accuracy approaches that of the GD optimisation*. Although the learned function is not as accurate as GD for optimisation, it does come close. The function learned from the training samples is learned with a high level of accuracy, but this can never be perfect for any data set. More importantly, if the potential sub-optimal geometries in the training sat are treated as noise in the data, it is evident that the SVM learns a function that improves on some if the initial data. On average, this produced geometries with a deflection under stress only 1.51% greater than those found by GD with a training set of 650. The variance in performance at this point is also low, representing a high degree confidence in these solutions.
- *The accuracy is within tolerances dictated by the manufacturing process*. The small shortcoming in performance of solutions predicted by the SVM becomes negligible when fabrication is considered. The error of the function measured in node point positions was found to be 1/5th the finest resolution of the stereolithography machine.
- *The learned function results in a smoother overall structure*. The avoidance of overfitting by a smoother learned function is beneficial both at the scale of the individual unit, and the whole structure. In the first instance, some predicted structures can actually perform better than what would be found by GD in instances where GD results in sub-optimal local optima. In the second instance, the overall combined structure benefits by a continuous functional estimation by producing a more gradual transition between adjacent unit cubes. This avoids potential weak points caused by recombining individually optimized structures.
- *The learned function is quicker for optimising larger structures*. Finding an optimal structural based on the learned function is far quicker than performing a full optimisation via gradient descent, as each sample of the latter requires a full finite element analysis, and one sample must be made for each dimension to calculate the gradient at each step. Learning the function for optimal structures however is time consuming, as in the example case studied, 650 fully optimized examples were required to learn the function at the outset. Many structural problems require the optimisation to be performed only once, but for those in which a similar structural optimisation is needed repeatedly, the initial investment in learning the function of optimal geometries can make the overall optimisation far more efficient. In the case of an object composed of many units of an arrayed topology as shown, the computation time becomes less for the learned function as the size of the object grows beyond 650 unit cubes. Larger sizes take an even greater advantage in time. As this method of optimisation is meant

to be scalable to ever-larger objects, the learned function represents a substantial advantage in speed.

When the problem is well defined, i.e. the environment and topology are constant and the loads can be quantified by a continuous valued vector, we have shown that it is possible to learn the function of optimal structures given the specified loading condition. Rather than optimisation by repeated sampling and evaluation of a physics-based model, it is thus possible to make design decisions for this structural problem based entirely on learning from previous examples. We have shown a method that uses this technique to predict optimal structures based on this principle, in which the training is performed in advance, and a structure is then produced that rivals the initial training set in strength. For structures of repeated units of the type we are considering, this method is many times more efficient than standard optimisation algorithms, and is thus a significant contribution to this problem of structural design.

The problem has been formulated as one of microstructures, comprised of a very large number of units with pre-defined topology but flexible geometry. The units used however, have been defined only relatively, and there is no reason in principle why such a technique could not be applied to structures of a larger size. As the training requires several hundred examples, the practical benefit of this approach in terms of speed is only evident when structures contain a number of units far greater than this, as do the microstructures we have been considering even of several centimetres. The rapid prototyping technologies used, however, are only part of a class of manufacturing methods including CNC cutting and milling that are being used at much larger scales. With recent architectural projects in excess of one kilometre and the enclosure of entire city neighbourhoods with space frame roofs becoming feasible, such an approach to optimisation may be valuable.

Most unexpected of the findings was that in generalising from the examples presented, the learning algorithm was so often able to actually outperform the original optimisations on which it was trained. Once trained on successful precedents, the machine, in a sense, knows intuitively what works based on its prior experience, and can then predict optimal structures that rival or even exceed the initial training set in strength. This is a result not of strict analysis however, but of inductive learning.

Acknowledgements

The authors would like to thank Dr. Joel Ratsaby and Prof. Bernard Buxton for their guidance and helpful suggestions in this work.

References

Adeli, H and Cheng, N: 1993, Integrated genetic algorithm for optimisation of space structures, *Journal of Aerospace Engineering* **6**(4): 315-328.

Arciszewski T and Ziarko W: 1990, Inductive learning in civil engineering: Rough sets approach, *Microcomputers in Civil Engineering* **5**:19-28.

Chen, YM: 2002, *Nodal Based Evolutionary Structural Optimisation Methods*, PhD Thesis, University of Southhampton.

Duda, RO, Hart, PE and Stork DG: 2001, *Pattern Classification*. John Wiley, NY.

Duffy AHB: 1997, The "what" and "how" of learning in design, *IEEE Expert: Intelligent Systems and Their Applications* **12**(3): 71-76.

Grzeszczuk, R, Terzpoulos, D and Hinton G: 1998, NeuroAnimator: Fast neural network emulation and control of physics-based models, *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 9-20.

Hanna, S and Haroun Mahdavi, S: 2004 Modularity and flexibility at the small scale: Evolving continuous material variation with stereolithography, *in* P Beesley, W Cheng and R Williamson (eds), *Fabrication: Examining the Digital Practice of Architecture*, University of Waterloo School of Architecture Press, Toronto, pp. 76-87.

Haroun Mahdavi, S and Hanna, S: 2003, An Evolutionary approach to microstructure optimisation of stereolithographic models, *Proceedings of CEC2003, The Congress on Evolutionary Computation*, Canberra, Australia, pp. 723-730.

Haroun Mahdavi, S and Hanna, S: 2004, Optimising continuous microstructures: A comparison of gradient-based and stochastic methods, *Proceedings of SCIS & ISIS 2004, The Joint 2nd International Conference on Soft Computing and Intelligent Systems and 5th International Symposium on Advanced Intelligent Systems*, Yokohama, Japan, pp. WE-7-5.

Kicinger R, Arciszewski T and De Jong K: 2005, Parameterized versus generative representations in structural design: An empirical comparison, *Proceedings of GECCO '05,* pp. 2007-2014.

Meiler, J and Baker, D: 2003, Coupled prediction of protein secondary and tertiary structure, *Proceedings of the National Academy of Sciences of the United States of America* **100**(21): 12105-12110.

Molecular Geodesics: 1999, Rapid prototyping helps duplicate the structure of life, *April 99 Rapid Prototyping Report*, Cyan Research Corporation.

Murawski K, Arciszewski T and De Jong K: 2000, Evolutionary computation in structural design, *Engineering with Computers* **16**: 275-286.

Murdoch T and Ball N: 1996, Machine learning in configuration design, *AI EDAM* **10**: 101-113.

Neocleous CC and Schizas CN: 1995, Artificial neural networks in marine propeller design, *IEEE Computer Society Press* **2**: 1098-1102.

Ping, Y: 1996, *Development of Genetic Algorithm Based Approach for Structural Optimisation*, PhD Thesis, Nanyang Technological University.

Quinlan JR: 1993, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Reich Y: 1997, Machine learning techniques for civil engineering problems, *Microcomputers in Civil Engineering* **12**: 295-310.

Reich Y and Barai SV: 1999, Evaluating machine learning models for engineering problems, *Artificial Intelligence in Engineering* **13**: 257-272.

Schoenhauer, M: 1996, Shape representations and evolution schemes, *Proceedings of the 5th Annual Conference on Evolutionary Programming*, MIT Press, Cabridge, MA, USA, pp. 121-129.

Schwabacher M, Ellman T and Hirsh H: 1998, Learning to set up numerical optimisations of engineering designs, *AI EDAM* **12**(2): 173-192.

Sischka, J, Hensel, M, Menges, A and Weinstock, M: 2004, Manufacturing complexity, *Architectural Design* **74**(3).

Suykens, JAK, Van Gestel, T, De Brabanter, J De Moor, B and Vandewalle, J: 2002*, Least Squares Support Vector Machines*, World Scientific, Singapore.

Szczepanik W, Arciszewski T and Wnek J: 1996, Emperical performance comparison of selective and constructive induction, *Engineering Applications of Artificial Intelligence* **9**(6): 627-637.

Vapnik V: 1995, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.

Von Buelow, P: 2002, Using evolutionary algorithms to aid designers of archictural structures, *in* PJ Bentley and DW Corne (eds) *Creative Evolutionary Systems*, Morgan Kaufmann, pp. 315-336.

Wang, LH, Liu, J, Li, YF and Zhou, HB: 2004, Predicting protein secondary structure by a support vector machine based on a new coding scheme, *Genome Informatics* **15**(2): 181-190.

LEARNING FROM "SUPERSTAR" DESIGNERS

PAUL A RODGERS
Napier University, UK

Abstract. Recent research has suggested that it is more important to study expert designers than novices. Typically, however, design expertise has been seen by researchers as the accumulation and organization of domain-specific knowledge. This work, however, views design expertise not only as knowledge and skills-based, but also as directly linked to the designer's critical and commercial success. This paper sets out to explore what makes six of the world's most distinguished and expert designers working today both critically and commercially successful? Moreover, the paper seeks to identify if they possess uniqueness in their genetic make up? Adopting the Watson - Crick Model of living organisms as a speculative model for each of the designer's "cultural DNA", this paper illustrates the significant design and cultural factors which the designers possess and exploit in their work.

1. Introduction

A number of recent papers have suggested that it is more important to study expert designers than novices as this will yield a deeper understanding of design thinking (Cross 2002; Cross and Edmonds 2003; Cross and Lawson 2005). Generally speaking, however, design expertise has been seen by the design research community as the collection and organization of domain-specific knowledge and skills of the designer. This paper, however, views design expertise not merely as knowledge and skills-based, but also as directly linked to the designer's critical and commercial acclaim. To this end, this paper provides an insight into the backgrounds, identities and working practices of a number of the world's most distinguished and expert designers working today. The six "superstar" designers presented here have been drawn from a number of design centres throughout the world and carefully selected on their current contribution to contemporary design practice and thinking (Rodgers 2004). Using traditional semi-structured interview techniques (Jordan 1998), the paper sets out to explore what makes these expert designers both critically and commercially successful?

J.S. Gero (ed.), Design Computing and Cognition '06, 583–601.
© 2006 *Springer. Printed in the Netherlands.*

Moreover, the paper seeks to identify if they have a unique methodology in their approach to design projects or whether they possess uniqueness in their genetic make up? Using the Watson - Crick Model of living organisms (Crick 1962) as a speculative model for each of the expert designer's "cultural DNA", this paper illustrates the significant design and cultural factors which the designers possess and utilize in their work.

2. Design Expertise

It is widely acknowledged that expertise is linked to domain-specific knowledge. Studies of domain-specific knowledge have included well-structured or ill-structured problem domains such as physics and mathematics (Larkin et al. 1980; Chi et al. 1981), and design (Akin 1979). All these studies show that detailed, domain-specific knowledge is necessary to solve problems successfully. Moreover, Ericsson and Lehmann (1996) discovered that the superior performance of experts is usually domain-specific, and does not transfer across domains. It has also been shown that experience plays a significant part in designer expertise. Empirical work has shown that there are a number of significant differences which distinguish experts from novices or non-experts (Glaser 1986; Badke-Schaub et al. 2001). These are differences with respect to the rapidity and accuracy of the general information process, differences in the manner that an expert or novice organizes their knowledge and the quality therein, and differences in the cognitive complexity of an individuals working memory. That is, experts build more complex knowledge and information representations than non-experts (Sternberg 1995).

2.1. DEFINITIONS

Design (di'zain) [vb.] means to work out the structure or form of (something), for example by making a sketch, outline, pattern, or plans. Expert ('ɛkspɚ:t) [n.] is defined as a person who has extensive skill or knowledge in a particular field. Expertise ('ɛkspɜ:ti:z) [n.], according to the dictionary definition (Hanks 1986), is described as possessing a special skill, knowledge, or judgment. Design expertise, as described by the dictionary then, is the skilful or expert preparation and execution of a plan. This definition does not fully address the complexities of the activity the design research community recognize as design, however.

3. Design Wisdom and the Notion of the "Superstar" Designer

Expertise in design commonly manifests itself throughout the design process in a whole manner of ways. For instance in the way that designers' research, analyze, challenge, (re)formulate and propose and present solutions to often

ill-defined problems. More specifically, designers draw upon and utilize their skills and knowledge of materials and manufacturing processes, their experiences of the market in which they operate, and their perceptual, visual-spatial knowledge (Eckert et al. 1999).

In recent studies, Cross has investigated the working practices of several designers from different domains including racing car design (Cross and Clayburn Cross 1996) and product design (Cross 2001). In his analyses of these designers, Cross (2002) believes there are three key design process elements in the working methods of exceptional designers. He lists these as:

1. Taking a broad 'systems approach' to the problem, rather than accepting a narrow problem definition and/or criteria;
2. 'Framing' the problem in a 'distinctive and personal' way;

   and
3. Designing from 'first principles', Figure 1.

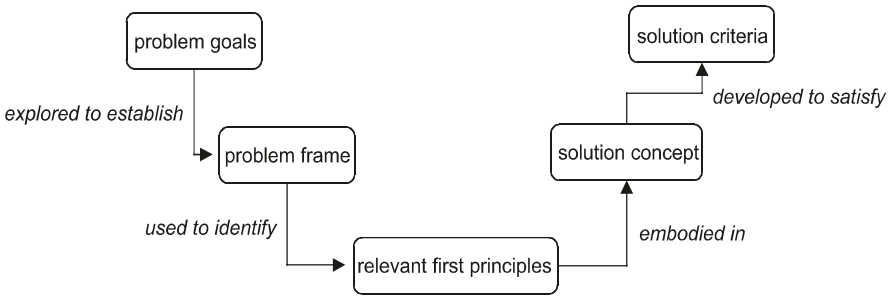


Figure 1. Cross' general design process model of 'exceptional' designers.

These 3 key elements, backed by Cross' empirical evidence, mirror many of the recommendations and suggestions of the 'early' design theorists such as Jones (1981), Pahl and Beitz (1984), and French (1985) and as such are largely not surprising.

The notion of 'design wisdom', as an extension of the accumulation and organisation of design knowledge and skills, has been challenged recently by Dudley (2000). In her work she cites a proverb for modern times attributed to the astrophysicist and author Clifford Stoll (Stoll 1996) who states:

> "data is not information, information is not knowledge, knowledge is not understanding, understanding is not wisdom."

Yet, in certain design research circles, the opposite of Stoll's proverb is utilised widely as an indication of what the design research community expects of its designers and design students through engaging at a sufficient level with their chosen subject. In terms of design education, we expect students to be able to transform data into information and through critical

practice and analysis, develop understanding and specialized knowledge of their chosen design path. One cannot guarantee, however, that students will possess the necessary expertise or wisdom upon graduating. As Dudley (2000) states:

> "Wisdom is attained by the experience of gaining knowledge and understanding, not only of one's subject, but also of oneself."

The designers presented here as "superstar" designers are well aware of their status and identity within the design world. They are commonly acknowledged as highly successful in their different disciplines, they have different educational backgrounds and personal experiences (i.e. architecture, design, and engineering) and they adopt differing approaches in their design practice (*i.e.* commercial, experimental, and critical). Moreover, it is speculated that continued reflection on design practice (Schon 1991) has provided designers with tools to design more than just architecture and products. That is, designers are designing themselves. In other words designers have created identities or "brands" for themselves often backed by corporations with which the public identifies. For instance, as Hollington (1998) states:

> "Starck may be the last great 'design hero', but he is also a brand."

Karim Rashid, interviewed and described later in this paper, is arguably the most recently "branded designer". Rashid's particular brand of modernist elegance has generated international acclaim, as well as the unofficial title of the design world's hippest jack-of-all-trades (Rashid 2001). His latest projects include new Emporio Armani Boutiques, several restaurants in New York City, and gallery installations. He also designs products, cosmetics, and fashion accessories for various international clients such as Issey Miyake, Zeritalia, Estee Lauder, Tommy Hilfiger, Giorgio Armani, Sony, Zanotta, Citibank, and others (Rashid et al. 2005).

4. Critical - Commercial Design Expertise

Design expertise from a critical, commercial perspective however moves forward the definition of the term 'expertise' to include a more holistic view of contemporary design success. That is, the term is meant to indicate more than the accumulation and organization of design-relevant knowledge and skills. Critical - commercial design expertise takes the accumulation of design knowledge and skills as a given, but necessitates evidence of both commercial success (*i.e.* sales success) and critical acclaim (*i.e.* amongst the designer's peers and design critics).

The author has recently completed a project interviewing a number of the leading design figures from cultural design centres around the world including London, New York, Amsterdam, Paris, and Tokyo. The designers

chosen are critically and commercially acclaimed on a global basis and feature in recent publications relating to contemporary design practice (Fiell and Fiell 2001; Terragni 2002).

5. Design Expertise as Cultural DNA

Cultural DNA, often referred to as "cultural capital" (Bourdieu 1984), exists within all people (including design practitioners, design students, etc.) and has been shown to be a major contributing factor towards the development of any designed artifact (Strickfaden et al. 2005).

Moreover, an individual's cultural identity is shaped largely by the customs, traditions, language, and superstitions that the individuals within the community utilize. Dawkins (1989) believes cultural transmission is analogous to genetic transmission in that, although basically conservative, it can give rise to a form of evolution. For example, art, architecture, engineering and technology all evolve in historical time in a way that looks like highly speeded up genetic evolution. Gell-Mann (1995) has developed the laws of natural selection and genetic diversity to cultural and linguistic phenomena and combined these concepts and labels into the amalgam "cultural DNA", first coined by Hazel Henderson (1999) in her analysis of global economic markets.

The importance of cultural frames of reference and influence and sources of inspiration for designers is well acknowledged and documented (Oxman 1990; Heylighen and Verstijnen 1997). Taking this acknowledgement as a starting point, the author set out to explore the "cultural DNA" of a number of the world's top designers using the Watson-Crick model (Crick 1962) of living organisms as a model wherein the structure of each "cultural DNA" is in the form of a double helix of two chains with each chain linked by a series of cultural connecting rungs. The goal being not merely to detect specific design influences, but rather to collect and classify some of the principal cultural ingredients from the worlds of art, design, cinema, and literature, etc. that successful contemporary designers use in their design activities.

6. Methodology

Approximately 30 of the world's most successful designers, from culturally significant centres throughout the world including New York City, London, Paris, Amsterdam, and Tokyo, were interviewed during this project. The interviews centred on what iconic influences and inspiration sources (Lawson 1994) the designers used to inform and drive their work. That is, what artists and designers do they admire, what books and magazines do they read, and what particular pieces from fine art, cinema or architecture are they influenced by.

The approach adopted here has been developed from the author's earlier work, in which he explored the relationship between undergraduate design student's degree performance (*i.e.* success) and their iconic and cultural inspiration sources (Rodgers and Milton 2001). Similarly, the aim of this work is to show the range and cultural diversity of influences that the world's top designers rely upon and use in their design work.

Each interview was conducted in the designer's normal work location, Figure 2. This is important as the designer can relax in his/her familiar surroundings and details of design projects and/or examples of inspiration are close at hand. As Cross and Lawson (2005) indicate, the majority of studies of outstanding designers have been based on interviews. Interviews give a richer picture than formalized data and also enable insights to emerge that may not have been planned by the interviewer. The major disadvantage of using interviews however is that they are very time consuming (e.g. post interview transcription, etc.).

Figure 2. Interviews with "superstar" designers in their studio (video stills).

6.1. MODEL

The genetic make up or DNA of living organisms (Crick 1962) is used as a speculative model in this paper to illustrate the similarities and differences of the "superstar" designers responses collected during the interviews. As aforementioned, each designer was interviewed in their own studio for approximately one to two hours.

The four stage procedure for modeling each designer's "cultural DNA" is shown in Figure 3. The first stage is the interview. Secondly, each interview was video taped and later transcribed (on average each interview transcript

ran to approximately 3000 words). Thirdly, each interview transcript contained a number of responses relating to the designers' influences which, in turn, were classified and categorized into one of the cultural bases of architecture (Arc), product design (Pd), cinema (C), art (A), literature (Lit), automotive design (Ad), music (M), or influential people (Ip). The number of responses collected and categorized for each designer during this project ranged from, at the lower end, around a dozen to over 100. Finally, in stage 4 each categorized response was modeled using the "cultural DNA" schema shown in greater detail in Figure 4.

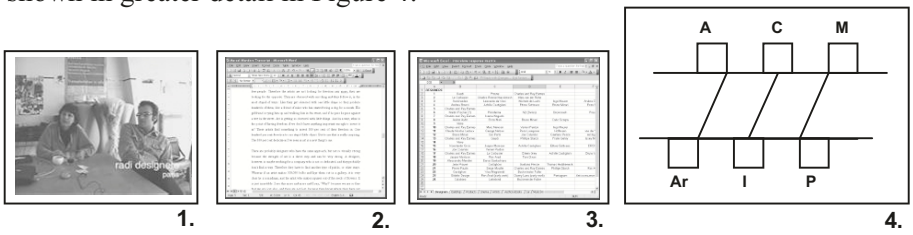


Figure 3. "Cultural DNA" four stage modelling process.

The structure of hereditary material (*i.e.* DNA) is made up of two chains coiled around one another in the form of a double helix with each chain linked by the purines[1] adenine (a) and guanine (g) and the pyrimidines[2] thymine (t) and cytosine (c). The adenine (a) strand always pairs with the thymine (t) on the other, whereas the cytosine (c) strand always pairs with guanine (g). The two strands are said to be complementary to each other. Thus a schematic representation example of an individual's DNA might read something like:

a g c t t a a g g c a t a c g c c g g t a a c g t a c c g g t t a c t a c a a c g t t
t c g a a t t c c g t a t g c g g c c a t t g c a t g g c c a a t g a t g t t g c a a

Similarly, the structure of each designer's "cultural DNA" make up is in the form of a double helix of two chains with each chain linked by a series of cultural connecting rungs, Figure 4. The two chains denote design influences from the bases of architecture (Arc), product design (Pd), cinema (C), art (A), literature (Lit), automotive design (Ad), music (M), and influential people (Ip). Again the strand pairs of Ad and Arc, C and Ip, M and Pd, and Lit and A are intended to be complementary to each other. Likewise, an example of a designer's DNA schematic representation might look something like:

[1] a derivative of purine; especially : a base (as adenine or guanine) that is a constituent of DNA
[2] a derivative of pyrimidine; especially : a base (as cytosine, thymine, or uracil) that is a constituent of DNA

Arc M Lit Lit A Pd C Ip C C M Lit Arc A Ad Ad Lit C M C Ip C Arc A M
Ad Pd A A Lit M Ip C Ip Ip Pd A Ad Lit Arc Arc A Ip Pd Ip C Ip Ad Lit Pd

The significant thing about each designer's "cultural DNA" is that it is completely unique. Although some of the designers in this study share common cultural elements in their responses to cinema influences or architectural icons no two "cultural DNA" chains are the same. This uniqueness and sharing of "cultural DNA" is further discussed in Section 8 of the paper.

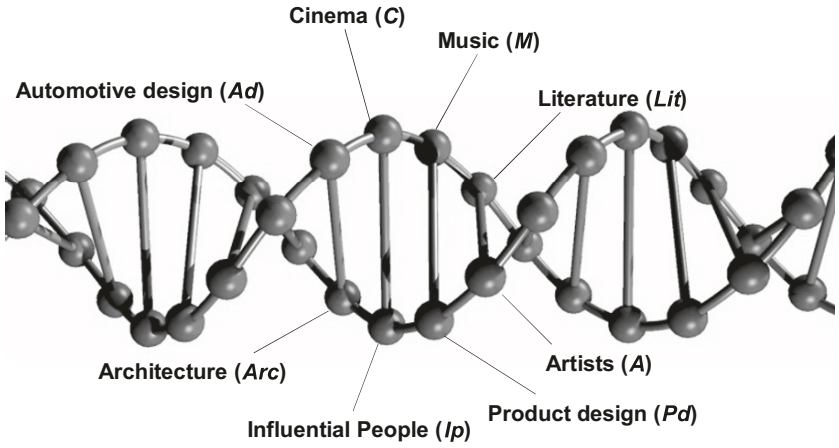


Figure 4. Cultural DNA schema.

7. "Superstar" Designer DNA

This section of the paper outlines a small but significant portion of the six designers' identities as "cultural DNA" from the perspective of their responses to questions regarding where their ideas originate, what influences their work and what inspires them from a cultural context. The goal is not to detect specific cultural influences, but rather to collect and classify the principal ingredients which successful contemporary designers utilize in their design work.

7.1. NICK CROSBIE (INFLATE DESIGN), LONDON

The philosophy of INFLATE Design is to design and produce original, fun, functional and affordable products. Launched in 1995, INFLATE exhibited their collection of inflatable products at 100% Design London and achieved a remarkable response. Recently, INFLATE have added a range of dipped PVC products to their portfolio as well as their inflatable products (Williams and Albus 1998).

Nick Crosbie of INFLATE's DNA specimen, Figure 5, shows his main influences to be the work of Charles Bukowski (Lit), the VW Beetle (Ad), Kubrick's 2001 (C), Future Systems' Media Tower at Lords Cricket Ground (Arc), the music of the Pet Shop Boys (M), and the pop artist Bridget Riley (A). Crosbie cited the work of Charles and Ray Eames as doubly influential, for both their furniture (Pd) and their enduring spirit (Ip).

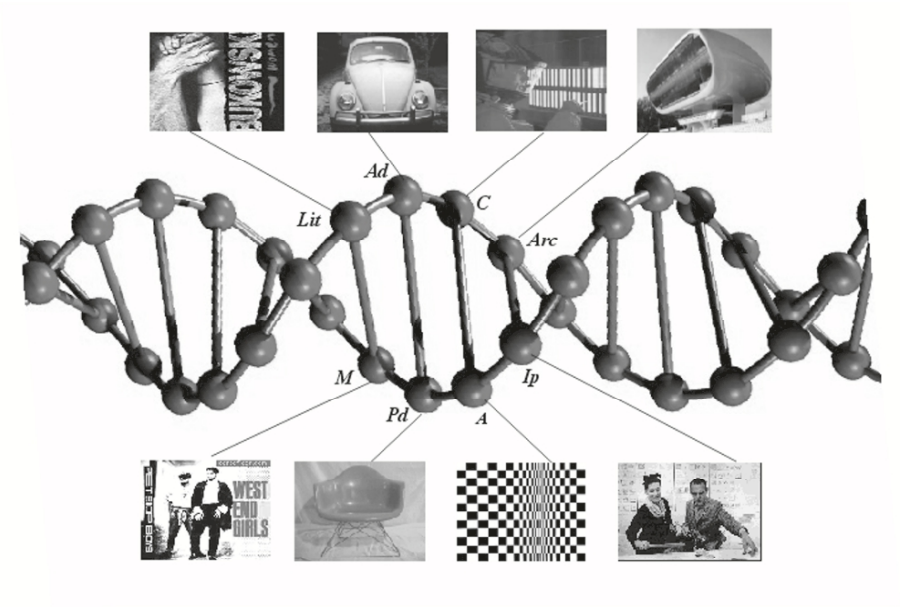


Figure 5. Nick Crosbie's cultural DNA specimen.

7.2. KARIM RASHID, NEW YORK CITY

Karim Rashid is one of the best-known and most prolific designers at work in the world today. Rashid works in diverse fields such as architecture, high-tech industrial design products, and cosmetics' packaging for clients such as Flos, Herman Miller and Sony. Rashid has coined the term "Sensual Minimalism" to describe much of his design work (Rashid 2001). Karim Rashid's influences, Figure 6, include the Mercedes Smart Car (Ad), the film Tron (C), French disco music (M), the work of Brancusi (A), the influential figure of Andy Warhol (Ip), Eero Saarinen's TWA Terminal in New York (Arc), Starck's phone for Philips Design (Pd), and Bret Easton Ellis' book American Psycho (Lit).

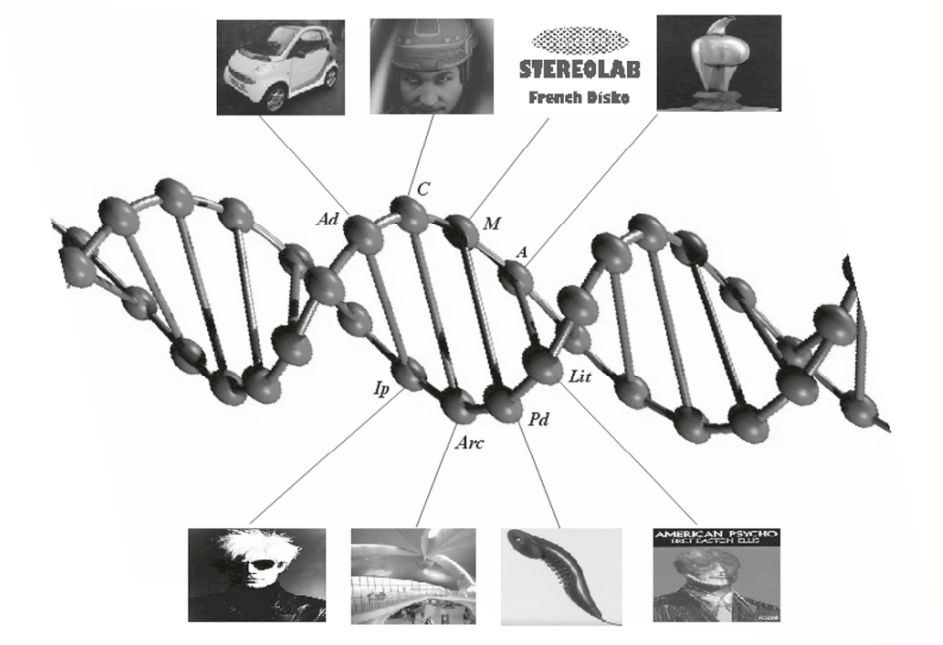


Figure 6. Karim Rashid's cultural DNA specimen.

7.3. RON ARAD, LONDON

Ron Arad originally studied architecture at the Jerusalem Academy of Art before coming to London to complete his architectural training at the Architectural Association in 1979. Today, Ron Arad is seen as one of the International superstars of design (Sudjic 1999). Arad is as famous for his architecture (*e.g.* Tel Aviv Opera House, Belgo Restaurant, London) as he is for his furniture and product design (*e.g.* Bookworm shelving, Tom Vac chair). Arad always seeks to challenge conventions in his work, yet prefers truth to sincerity. Arad believes that Bob Dylan (one of his key iconic influences in Figure 7) sums this feeling up best when he says:

> "…to live outside the law you have to be honest." (Guidot and Boissiere 1997)

Ron Arad is influenced greatly by the work of Issey Miyake (Ip). Arad also cites the work of Jacques Tati as influential in his personal development as a designer, particularly Tati's film Playtime (C). He also includes Le Corbusier's masterpiece Notre Dame du Haut, Ronchamp (Arc), the work of Marcel Duchamp (A), Charles and Ray Eames' furniture (Pd), and the 1950's Fiat 500 (Ad) as major contributors to his work, Figure 6. The music

of Bob Dylan (M), mentioned earlier, is important to him as is the author Philip Roth (Lit).

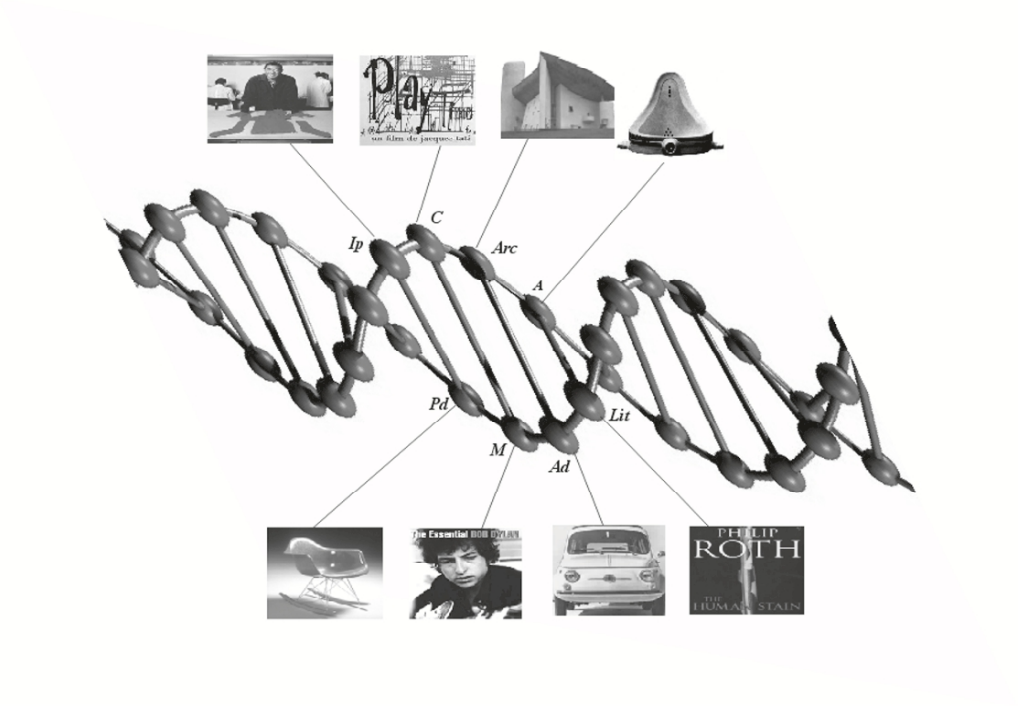


Figure 7. Ron Arad's cultural DNA specimen.

7.4. SCOTT HENDERSON, SMART DESIGN, NEW YORK CITY

SMART design's Director of Industrial Design Scott Henderson has an approach to design which he terms "expression sessions" which exploits the power of spontaneous thinking. His work has been included in the ID magazine's Annual Design review five times and has won a number of International Design awards. SMART produce a wide range of products for clients such as Black and Decker, Hewlett-Packard and Timberland.

Scott Henderson mentions the E Type Jaguar as a strong cultural icon in his make up (Ad). The work of Charles and Ray Eames (Ip), like Nick Crosbie of INFLATE, Ron Arad and Marcel Wanders, is listed by Henderson as important. The Seagram building in Manhattan, New York (Arc), Hitchcock's movies especially North by Northwest (C), and The Fountainhead by Ann Rand (Lit) are also mentioned in his cultural icons, Figure 8. Henderson cites a specific work (i.e. Tom Vac Chair) of Ron Arad as particularly important to him (Pd). He also lists the work of Brancusi (A),

which he shares with Karim Rashid as meaningful, and the Prodigy as his musical influence (M).

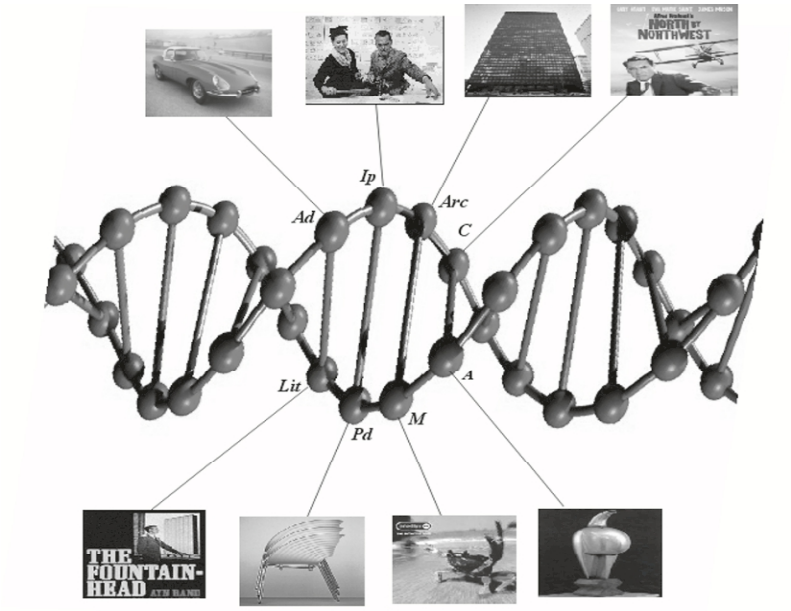


Figure 8. Scott Henderson's cultural DNA specimen.

7.5. FLORENCE DOLÉAC AND OLIVIER SIDET, (RADI DESIGNERS), PARIS

Florence Doléac and Olivier Sidet view the process of design as one of interpolation and transfiguration, rather than of merely representation. In other words, design is not about the interpretation of the meaning of an object only, but rather about the many possibilities of its interpretation. This is best summarised when RADI state (Fiell and Fiell 2001):

> "By transposing our philosophy of design into forms that are at once humorous and subtly out-of-step but also thoroughly humanised, we try to project a gentle yet profound way of imagining tomorrow."

The group members of RADI work together on a variety of projects covering product, exhibition and interior design. RADI's clients include Air France, Issey Miyake, Cartier and Schweppes. In terms of their "cultural DNA", RADI list Gabriel Orozco, the Spanish Artist (A), Frank Gehry's furniture (Pd), and Umberto Eco's work (Lit) as potent icons in their make up. They admire the work of Salvador Dali (Ip), the VW Beetle (Ad), which they have in common with Nick Crosbie of INFLATE, and the Guggenheim Museum, Bilbao (Arc). RADI state that the work of Stanley Kubrick, most notably the film 2001 (C) which they also share with Nick Crosbie of

INFLATE, and the music of Beck (M) amongst others are highly influential to their work, Figure 9.

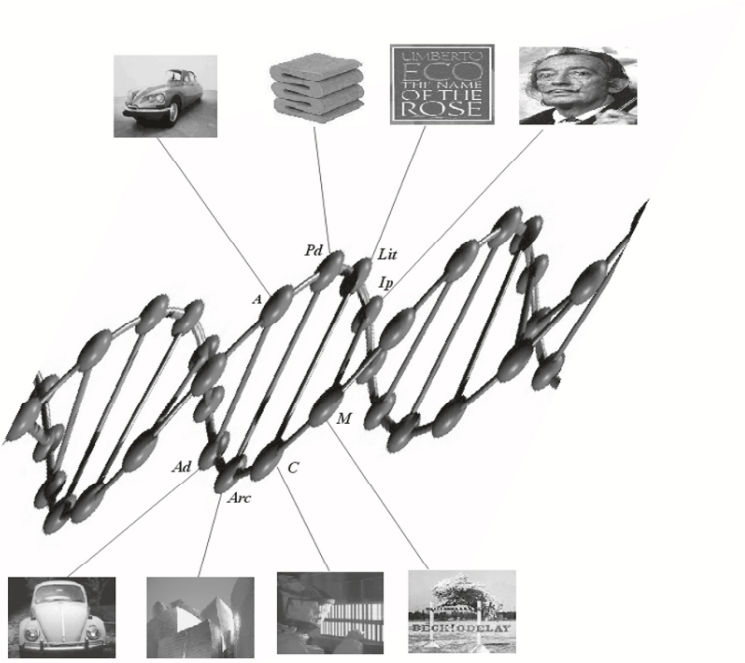


Figure 9. Florence Doléac and Olivier Sidet's cultural DNA specimen.

7.6. MARCEL WANDERS, AMSTERDAM

Marcel Wanders' work, according to Andrea Branzi quoted in Joris (1999), is:

> "…placed within that range of researches that investigate about a new relationship between technology and nature."

This is best exemplified in Wanders' "Dry Tech Knotted Chair" where he opted to integrate macramé, a traditional way of working, with Dry Tech experiments (air and space technology) with new materials at the Delft University of Technical Engineering (van Zijl 1997). Marcel Wander's "cultural DNA" specimen, Figure 10, comprises the work of Tony Cragg (A), Le Corbusier's Notre Dame du Haut, Ronchamp (Arc), the music of George Michael (M), and Ken Wilber books (Lit). Wanders is a huge admirer of Porsche cars, particularly the late 1980's 928 model (Ad), the design philosophy of Swatch (Pd), and all of the Rocky movies (C). Wanders cites the work of Charles and Ray Eames as influential in his work

(Ip). This final cultural base he shares with both Nick Crosbie (INFLATE) and Scott Henderson (SMART design).

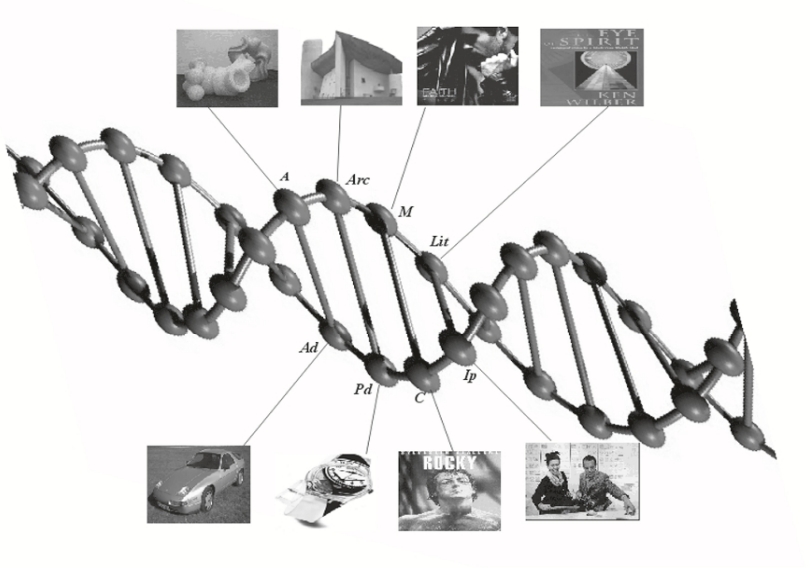


Figure 10. Marcel Wanders' cultural DNA specimen.

8. "Cultural DNA" Uniqueness

This paper set out to identify whether or not "superstar" designers possess uniqueness in their genetic make up? Each of the six designers "cultural DNA" specimens have been studied using the Watson-Crick model of living organisms as a speculative model for designer expertise.

The results from the interviews indicate that there are a number of confluent "cultural DNA" elements which the designers share. Using the "cultural DNA" schema illustrated earlier, Table 1 highlights these shared DNA elements. For instance, the base Arc (architecture) comprises important buildings such as Foster's Hong Kong and Shanghai Bank, Future Systems' Media Tower, and the Seagram Building in New York, but Le Corbusier's Ronchamp is a cultural element which is shared by both Ron Arad and Marcel Wanders.

Within the base Pd (product design), there is a variety of responses which illustrates the diversity of skills, knowledge and expertise that the designers possess. As most of the designers' work is three-dimensionally-based this is not at all surprising. An interesting outcome of this base, however, is the significance of the work of Charles and Ray Eames. Their work, in particular, is cited frequently both within the Pd base and the Ip (influential personalities) base.

The cinema base (C) contains other significant cultural connection points for many of the designers interviewed during this project. The notable movies cited as influential cultural reference points include Ridley Scott's "Bladerunner", Stanley Kubrick's "2001", Alfred Hitchcock's "North by Northwest", and the works of Andre Tarkovsky, Akira Kurosawa and Peter Greenaway. Stanley Kubrick's 2001 is shared by Florence Doléac and Olivier Sidet (RADI designers, Paris) and Nick Crosbie (INFLATE, London).

TABLE 1. "Cultural DNA" specimen comparison.

	Arc	Pd	C	A	Ad	Ip
Nick Crosbie						
Karim Rashid						
Ron Arad						
Scott Henderson						
Florence Doléac & Olivier Sidet						
Marcel Wanders						

The art base (A) elements of the "cultural DNA" reflected a wide range of specific art disciplines including painting, sculpture, video art and conceptual art. Specific artists named as influential included the Japanese sculptor Isamu Noguchi, the Japanese video artist Mariko Mori, Pop Art protagonists such as Andy Warhol and Bridget Riley, and Pablo Picasso. Both Karim Rashid and Scott Henderson (SMART design, New York City) cited the work of Brancusi as one of their most important A base DNA elements.

The DNA connecting node of literature (Lit) typified most the cultural boundaries between the designers in that the literary influences reflected the cultural experiences and education of the designer's situation. That is, designers based in the USA stated well known USA/Western European authors and books as their major influences as did UK-based designers to a lesser extent. Designers based in Paris (*e.g.* Florence Doléac and Olivier

Sidet) or Amsterdam (*e.g.* Marcel Wanders) listed non-English language literary works as inspirational to them.

In terms of automotive design inspiration, the Ad shared DNA bases comprised a major automobile design classic, namely the original Volkswagen Beetle. Again Florence Doléac and Olivier Sidet (RADI, Paris) and Nick Crosbie (INFLATE, London) shared a common base here with their Beetle response as they do with their cinematic (C) base. Other notable stated Ad influences include classic cars from the major manufacturers Citroen, Mercedes and Porsche.

The cultural DNA base M (music), like Lit, highlights a wide range of responses. Unlike Lit, however, these responses do not mark out the cultural boundaries of the designers themselves. Rather they illustrate that musical influences (M) cross borders *(i.e.* Dutch designers citing English musicians and USA designers stating French musicians as influential). An interesting outcome of the M base responses is that although the replies are not geographically dependent, they appear to be temporally dependent amongst the designers interviewed.

In summary, the "strong" cultural DNA totems (from the bases of Arc, Pd, C, A, Lit, Ad, M, and Ip) associated with the six designers selected and presented here are Charles and Ray Eames, Le Corbusier (Arc, Pd, Ip), Stanley Kubrick's 2001 (C), the work of Brancusi (A), and the VW Beetle, (Ad). It is interesting to note from Table 1, that music (M) and literature (Lit) are the only bases where there is no confluence amongst the designers. From the research carried out, the Lit base appears to be geographically dependent (i.e. French designers citing French writers as influential and so on) whereas the M base appears to be temporally dependent (*i.e.* the age of the designer reflects directly their taste in musical influences, such as Ron Arad stating Bob Dylan, and Nick Crosbie listing the Pet Shop Boys).

The high incidence of cultural icon confluence amongst the designers interviewed here (*i.e.* more than 25% of the total responses are shared by more than one designer) appears to lend weight to Featherstone's notion of *"polyculturalism"*. That is, that due to the increasing international flows of money, products, people, images and information *"third cultures"* have developed which mediate between traditional national cultures (Featherstone 1990). Furthermore, according to Votolato (1998):

> "…design has become international…and the international nature of design practice have tended to standardize the design of goods, environments and the presentation of services around the world."

9. Conclusions

This paper has described the results of a project which is investigating where designers' ideas originate, what influences their work and what inspires them from a cultural context. The paper details a selection of six of the world's top designers' responses to prompts relating to their personal cultural icons. The aim here is not to detect specific design influences, but rather to collect and attempt to classify some of the principal cultural ingredients that successful designers utilise in their design and development activities. To this end, the paper has described the personal influences and the important iconic references of each designer involved. The main finding of the work is that many of the designers interviewed share "cultural DNA". This is interesting bearing in mind the fact that the designers are from different cultural centres, have different educational backgrounds and personal experiences and also adopt differing approaches in their design practice.

It appears that "cultural DNA" exists within all designers and is a major contributing factor towards the development of a designed artifact. It appears that designers and even "superstar" designers use their "cultural DNA" knowingly, unknowingly, creatively and spontaneously throughout their designing. That is, the designed artifact is 'born' in an ecosystem that contains other designed artifacts, and the experiences surrounding people's interfaces with the designed world of objects, places and spaces. In the case of design, those objects and experiences relate to the everyday lives and cultures of designers.

Acknowledgements

The author would like to express his gratitude to all the designers who contributed towards, and continue to support, this work.

References

Akin, O: 1979, *Models of Architectural Knowledge*, PhD Thesis, Carnegie-Mellon University, Pittsburgh, USA.

Badke-Schaub, P, Stempfle, J and Wallmeier, S: 2001, Transfer of experience in critical design situations, *in* S Culley, A Duffy, C McMahon and K Wallace (eds), *Design Management – Process and Information Issues (Proceedings of ICED 2001)*, Professional Engineering Publishing, London, pp. 251-258.

Bourdieu, P: 1984, *Distinction–A Social Critique of the Judgment of Taste*, Routledge, London.

Chi, MTH, Feltovich, PJ and Glaser, R: 1981, Categorisation and representation of physics problems by experts and novices, *Cognitive Science* **5**: 121-152.

Crick, FHC: 1962, The genetic code, *Scientific American* **207**(4): 66-74.

Cross, N and Clayburn Cross, A: 1996, Winning by design: The methods of Gordon Murray, racing car designer, *Design Studies* **17**(1): 91-107.

Cross, N: 2001, Achieving pleasure from purpose: The methods of Kenneth Grange, product designer, *Design Studies* **4**(1): 48-58.

Cross, N: 2002, Creative cognition in design: Processes of exceptional designers, *in* T Hewett and T Kavanagh (eds)*, Creativity and Cognition*, ACM Press, New York, USA.

Cross, N and Edmonds, E (eds): 2003, Expertise in design*, Design Thinking Research Symposium 6*, University of Technology, Sydney, Australia.

Cross, N and Lawson, B: 2005, Studying outstanding designers, *in* JS Gero and N Bonnardel (eds), *Studying Designers'05*, Key Centre of Design Computing and Cognition, University of Sydney, pp. 283-287.

Dawkins, R: 1989, *The Selfish Gene*, OUP, Oxford.

Dudley, E: 2000, Intelligent shape sorting, *in* E Dudley and S Mealing (eds), *Becoming Designers: Education and Influence*, Intellect Books, Exeter, England, pp. 53-62.

Eckert, C, Stacey, M and Wiley, J: 1999, Expertise and designer burnout, *in* U Lindemann, H Birkhofer, H Meerkamm and S Vajna (eds), *Proceedings of ICED 1999*, Technische Universitat Munchen, Munich, pp. 195-200.

Ericsson, KA and Lehmann, A: 1996, Expert and exceptional performance: Evidence on maximal adaptations on task constraints, *Annual Review of Psychology* **47**: 273-305.

Featherstone, M: 1990, *Consumer Culture and Postmodernism*, Sage Publications, London.

Fiell, C and Fiell, P: 2001, *Designing the 21st Century*, Benedikt Taschen Verlag, Cologne.

French, MJ: 1971, *Engineering Design: The Conceptual Stage*, Heinemann, London.

Gell-Mann, M: 1995, *The Quark and the Jaguar: Adventures in the Simple and the Complex*, Abacus, London.

Glaser, R: 1986, On the nature of expertise, *in* F Klix and H Hagendorf (eds), *Human Memory and Cognitive Capabilities*, Elsevier, Amsterdam.

Guidot, R and Boissiere, O: 1997, *Ron Arad*, Dis Voir Publishers, Paris.

Hanks, P (ed): 1986 (2nd ed), *Collins Dictionary of the English Language*, William Collins and Sons, Glasgow.

Henderson, H: 1999, *Beyond Globalization: Shaping a Sustainable Global Economy*, Kumarian Press, Bloomfield, USA.

Heylighen, A and Verstijnen, IM: 1997, Exposure to examples: Exploring CBD in architectural education, *in* JS Gero (ed), *Artificial Intelligence in Design 2000*, Kluwer Academic, Dordrecht, The Netherlands, pp. 413-432.

Hollington, G: 1998, The usual suspects, *Design*, Summer, pp. 62-63.

Jones, JC: 1992, *Design Methods*, Van Nostrand Reinhold, New York.

Jordan, PW: 1998, *An Introduction to Usability*, Taylor and Francis, London.

Joris, YGJM (ed): 1999, *Wanders Wonders: Design for a New Age*, 010 Publishers, Rotterdam.

Larkin, J, McDermont, J, Simon, DP and Simon, HA: 1980, Expert and novice performance in solving physical problems, *Science* **208**: 1335-1342.

Lawson, B: 1994, *Design in Mind*, Butterworth Architecture, London.

Oxman, RE: 1990, Prior knowledge in design: A dynamic knowledge-based model of design and creativity, *Design Studies* **11**(1): 17-28.

Pahl, G and Beitz, W: 1995, *Engineering Design: A Systematic Approach*, Springer-Verlag, Berlin.

Rashid, K: 2001, *I Want to Change the World*, Universe Publishing, New York, NY, USA.

Rashid, K, Bangert, A and Morgan CL: 2005, *Digipop*, Benedikt Taschen Verlag, Cologne.

Rodgers, PA and Milton, AI: 2001, What inspires undergraduate design students?, *The Design Journal* **4**(2): 50-55.

Rodgers, PA: 2004, *Inspiring Designers*, Black Dog Publishing, London.

Schon, DA: 1991, *The Reflective Practitioner: How Professionals Think in Action*, Ashgate Arena, London.

Sternberg, RJ: 1995, Expertise in complex problem solving: A comparison of alternative conceptions, *in* P Frensch and J Funke (eds), *Complex Problem Solving: The European Perspective*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 295-321.

Stoll, C: 1996, *Silicon Snake Oil*, Pan Books, New York.

Sudjic, D: 1999, *Ron Arad*, Laurence King Publishing, London.

Strickfaden, M, Heylighen, A, Rodgers, PA and Neuckermans, H: 2005, The 'culture medium' in design education, *in* PA Rodgers, L Brodhurst and D Hepburn (eds), *Crossing Design Boundaries*, Taylor and Francis, London, pp. 59-63.

Terragni, E (ed): 2002, *Spoon*, Phaidon Press, London.

van Zijl, I: 1997, *Droog Design 1991-1996*, Centraal Museum, Utrecht.

Votolato, G: 1998, *American Design in the Twentieth Century*, Manchester University Press, Manchester.

Williams, G and Albus, V: 1998, *Inflate*, Verlag Form, London.

THE IMPROVEMENT OF DESIGN SOLUTIONS BY MEANS OF A QUESTION-ANSWERING-TECHNIQUE (QAT)

CONSTANCE WINKELMANN AND WINFRIED HACKER
Dresden University of Technology, Germany

Abstract. In two experimental studies, the influence of question-based reflection on the quality of design solutions was investigated. The participants, experts with different know-how and professional experience, had to design an artefact that should meet a list of requirements. Subsequently, they were asked to answer non-product-specific questions with the opportunity to modify or revise their design. These interrogative questions (e.g. why, what for, how, where, etc.) aim at the semantic relations in systems, for example, causal, final, temporal, local, conditional relations, which are to be inferred in order to develop a mental representation of a system. These questions cause the participants to explain, justify and evaluate their finished design. We found significant solution improvements in both target groups, whereby the improvements of experts with a lower work-experience (job beginners) were significantly higher as those with higher experience.
This testifies to the effectiveness of question-based reflection in the early phases of design processes. The question-answering-technique is a useful tool for the systematic analysis and improvement of design solutions as well as for the optimization of design processes. We recommend utilising this technique in the training of engineering designers.

Possibilities of supporting the intuitive-creative draft sections in design problem solving examined so far, refer primarily to sketching or modelling as possible forms of externalisation (Pahl and Beitz 1997), the use of tools, the use of analysis methods or draft methodologies.

Since design activities as cyclic, iterative processes consist of both draft and evaluation sections, not only the support of draft sections is important but also of the intuitive evaluation sections, e.g. by using questions techniques (Ahmed et al. 2000; Kluge 1999). Studies by Wallace and Ahmed (2003) have shown that for around 90% of information requests designers contacted another person and that novices failed to ask the right questions.

J.S. Gero (ed.), Design Computing and Cognition '06, 603–618.
© 2006 *Springer. Printed in the Netherlands.*

In principle, differences between questions which provide information by another person or using documents (Wallace and Ahmed 2003), questions which support the decision making process (Eris 2003) and questions used as an instrument for the self-management, e.g. for methodical purposes, have to be made. The latter form the topic of this article. They can be furthermore divided into questions as a requirements check, thus for memory of the given demands of the order, and into questions which obtain mental impetuses by stimulating the production of semantic relations.

Questions form the catalyst of reflection: "Questions are thus one of the most basic and powerful elements of the reflection experience" (Daudelin 1996, p. 42). The kind of the questions used depends on the respective kind of the reflection process. According to Daudelin (1996), questions asking for the "what" stimulate a complete description of the situation. In contrast, questions asking for the "why" are useful for the analysis of the task or the result. Finally, questions asking for the "what" or "how" promote the development of new perspectives. These questions as suggested by Daudelin are considered interrogative questions. In contrast to yes- or no-questions which only ask for the existence or non-existence of circumstances, the interrogative questions ("why", "for what", "how", "with what"...) stimulate specific reconsideration in form of reasoning and evaluation of a problem which goes beyond pure descriptions (Doerner 1998). This is possible because the interrogative questions aim at the exhaustive dealing with so-called semantic relations. Thus, asking why-questions, for example, provokes the specific recourse to knowledge of causes, discernible from used causal conjunctions like "because" or "as". However, questions asking for the "what for" aim at the knowledge of the purpose and use of circumstances which might be identified by means of final conjunctions such as "in order to" and "with it". The goal of the explanation-, reason- and evaluation-evoking question technology is the stimulation of a reflective examination of the own draft solution.

Following the investigations of Ericsson and Simon (1993) or Bartl and Doerner (1998), not only the pure commentating and describing of processes but also the request for providing explanations and reasons are responsible for the increase in achievements. These will lead to meta-cognitions which allow a more effective planning of the problem-solving process, like the concentration on important aspects of the problem. Consequently, they are accompanied by an improved problem-solving achievement (Dominowski 1990). Ericsson and Simon (1993) see the success in the stimulation of a systematic change of the information-processing procedures.

The positive effects of question-based reflecting reported in the literature (Cottrell 2003; Daudelin 1996; Strzalka and Strzalka 1986) could be proven in recent investigations by Winkelmann et al. (2003) and Hacker and Wetzstein (2004).

What is considered a significant distinction is whether these techniques are used as process-interrupting means (Bartl and Doerner 1998) or after the completion of a draft process or draft sections (Strzalka and Strzalka 1986). In this respect, these techniques differ from activity-accompanying reflecting on the own thought process during the preparation and likewise from process-accompanying commentating in the sense of the method of loud thinking (Akin 1978; Eastman 1969). The use of reflection-stimulating questions after the completion of the draft suggests advantages because it does not work on a process-interrupting basis but stimulates a systematic and complete control – similar to a check list - and beyond that to an analyzing, reflective examination with developed (intermediate) solutions.

In an earlier study (Hacker and Wetzstein 2004), the effect of question-based as well as reflective examination with the finished draft solution on its quality was examined with students (N = 60) who had not had any prior design-methodical knowledge. They had to sketch an object of everyday use which should fulfil specific requirements. After finishing the draft, they were asked to describe, justify and evaluate their solutions on the basis of product-specific interrogative questions, Table 1, asked by a naive questioner.

This is a person without any expert knowledge who does not give any helpful information concerning the solution during the question-answering process. Meanwhile, the participants of the control group had to work on a questionnaire as to the well-being of themselves. After that the participants of both the experimental and control group had the possibility of revising their solutions. The results show significantly higher improvements in the quality of solutions within the experimental group in the comparison to the control group. Besides, in the experimental group significantly more participants developed new principles and added new explanations of functions to their design, whereas the participants of the control group made mainly corrections.

Another investigation (Winkelmann et al. 2003) compared five student groups (N = 150) doing the same task. Four groups were presented with product-specific questions after finishing the draft which had to be answered in different ways (silently, aloud without and with a partner, in writing). The fifth group received non-product specific questions (which had to be answered in writing only) which were used for the first time. As a result, there were significant improvements in the quality of the solution in each of the five conditions, independent of the kind of answering. Of crucial importance in the sense of an economically efficient support of draft processes is the improvement of the quality of solution when non-product specific questions are used. Thus, it allows the transfer of the question-answering technology to other objects.

In one of the most recent investigations (Winkelmann 2005), the effect of non-product-specific questions was proven when the questions had to be answered silently, aloud without and with partner. In all intervention groups the participants (N = 150 students) could improve the quality of their draft

solutions significantly. The mode of answering had no influence on the extent of improvement.

For reasons of explanation, the foremost question is in which specific way the designers improved the design. We therefore developed a system of categorization, Table 2, (Winkelmann 2005).

TABLE 1. Product-specific interrogative questions.

1. Which requirements should the grill meet?
Requirements of the single devices:
Stator
2. What is the position of the grill like? Is the grill standing stably?
3. Why did you decide on this variant? What are the advantages?
4. What are the disadvantages?
Grid and coal pan
5. What does the base of the grill look like and how are grid and coal pan fixed in it?
6. How can the grid and coal pan be removed? How can they be reinserted?
7. Why did you decide on this variant? What are the advantages?
8. What are the disadvantages?
Adaptation mechanism
9. How is the mechanism working?
10. How do you lock the mechanism?
11. How did you make sure that nothing cants?
12. Why did you decide on this variant? What are the advantages?
13. What are the disadvantages?
14. What would you like to modify?
15. Which effect and consequences would these modifications have?

Mainly supplementary changes either in written or graphic form were made. However, only 24% led to an improvement of the solution, whereas the innovative modifications like error corrections, further and new developments brought about a 82%-increase in the solution quality. Both, supplementary and innovative changes led in 63% of the cases to an increase in the solution quality. Novices made significantly more innovative changes, and in detail, they corrected more errors than experts.

TABLE 2. Type and number of modifications (N = 137).

	Frequency in %	
Type of modification	**Categorization (fine)**	**Categorization (rough)**
description	29.9	17
supplementation / graphically: relevant	17.5	64
irrelevant	37.2	
supplementation / in writing: relevant	19.0	
irrelevant	39.4	
correction of mistakes (error correction)	17.5	19
further developments	9.5	
new principles	7.3	

A second question is in which specific way the QAT influences the cognitive behaviour of the designers. For this reason, we conducted a further study (N = 84, Winkelmann 2006) and divided the QAT into two parts: a) into questions which aim at controlling the consideration of all requirements, and b) into questions which require a description, a justification and an evaluation of the own solution. Beyond that, the complete question catalogue was used. In the first condition the participants had to answer interrogative questions as to the requirements (memory questions) of the task (e.g. What is explicitly required for device A in the particular task?). In the second condition the participants were asked to describe, justify and evaluate their solutions on the basis of interrogative questions (thinking questions) (e.g. Why did you decide on this variant? What are the advantages? What are the disadvantages?). In the third condition the participants received both memory and thinking questions, i.e. the complete question catalogue. The interventions led to significant improvements in the solution quality and confirm the positive effect of the question answer technology. Significant differences between the question variants could not be proven, yet tendentiously the complete version of the question catalogue significantly led to more improvements compared with the two incomplete catalogues. The common effect of both kinds of questions is about ten times larger than solving the problem without any given questions. The results show that both questions which aim at controlling the requirements of the task as well as questions, requiring a description, a justification and an evaluation of the own solution significantly improve the solution quality. Therefore, the two parts of the complete question system might open up different, supplementing sources of the improvement of the solution quality.

In addition, we conducted an analysis of the statements of the respondents (N = 94) which means we analysed the given answers (Winkelmann 2006). On the basis of the used conjunctions as "relation-indicating words", the semantic relations were to be determined. Since the questions aim at both describing and justifying as well as evaluation one's own solution(s), primarily such conjunctions have been of interest which could be used by the participants. Thus, the request to explain and describe suggests the possible use of explicative and instrumental conjunctions; furthermore, forming causal and final relations is demanded by justifying. Evaluating one's own solutions contains, among other things, a weighing of pros and cons, which is particularly shown by the use of conditional, consecutive, concessive and adversative conjunctions. The search for alternative solution variants is brought about ideally by the use of alternative conjunctions, Table 3.

TABLE 3. Considered conjunctions and its meaning.

Conjunctions	Meaning	Examples
explicative	describing, explaining	so, therefore, e. g.
instrumental	means	so that, by
causal	cause	as, because, since
final	purpose, object, aim	with it, by it, that, to, in order to
conditional	condition	if ... so, in case, so far as
consecutive	consequence, result	so that, in order to, so, therefore, thus, consequently
concessive	comparison, comparative restriction	though, although, even though, even if
adversative	opposition	but, however, still
alternative	choice	or rather, or ... respectively, either ... or, or

The results show that persons without improvements used extensively more instrumental and explicative conjunctions than persons with improvements, Table 4. Translated into the describing, justifying and evaluating effect of the assigned questions, this means that persons without improvements remained significantly more on/at the describing level. In addition, a noteworthy difference concerning the used final conjunctions exists: they were considerably more frequently used by respondents with improvements. Further differences could not be proven.

The subject of the investigation which is presented in the following is the transferability of the question technology on specialists with a different degree of design-methodical experience. The expert's degree is of high

importance for measuring the achievements in design-problem solving. The comparison of experts and novices showed both qualitative and quantitative differences in reflection (Ertmer and Newby 1996). Rhone et al. (2001) – following Schoen's (1996) 'Reflective Practitioner' - found that experts form a larger problem area compared with novices. In addition, experts performed more iterative activities than novices (Adam et al. 2003; Atman and Turns 2001). These increased iterations correlated positively with successful solutions. Halton and Smith (1995) could show - as an explanatory contribution - that reflection of experts contained the deeper structure of problem-related solutions, whereas novices were more oriented towards surface characteristics.

TABLE 4. Mean and standard error of respondents with and without improvements producing conjunctions.

Stimulated mental activites	**Conjunctions**	**Respondents**			**t**	**Significance**
		without improvements		**with imrpovements**		
		n = 36		**n = 18**		
explaining	explicative	0.64 ± 0.05	>	0.47 ± 0.06	2.03	< .05
	instrumental	0.28 ± 0.04	>	0.16 ± 0.04	2.01	< .05
justifying	causal	0.56 ± 0.05	<	0.57 ± 0.08	-0.05	n. s.
	final	0.61 ± 0.07	<	0.85 ± 0.13	-1.80	< .05
evaluating	conditional	0.40 ± 0.05	>	0.32 ± 0.04	1.02	n. s.
	consecutive	0.77 ± 0.07	>	0.75 ± 0.07	0.18	n. s.
	concessive	0.37 ± 0.05	>	0.34 ± 0.04	0.42	n. s.
	adversative	0.16 ± 0.03	>	0.12 ± 0.02	1.04	n. s.
	alternative	0.22 ± 0.03	>	0.12 ± 0.03	1.25	n. s.

The assumption that experts always look for the simplest or an easier way when solving problems than novices could not be confirmed (Holyoak 1991). "In some ways, therefore, creative experts treat problems as 'harder' problems than novices" (Cross 2001). When solving similar tasks, creative experts orientate themselves more to previously found solution principles

than to solutions. "... such features of creative expertise can be seen in the strategies of outstanding expert designers " (Cross 2001).

However, Birch (1975) could show that problem solving is not exclusively based on insights, but it is dependent on former experiences. Goeker (1997) supports this result; according to his investigations experts proceed in a less abstract-thinking manner when solving design problems, but orientate themselves rather to their own experiences.

The hidden risk which is involved when one exclusively orientates oneself to existing knowledge and its modification according to the new problem situation exists in the insistence on earlier experience, whereby creative problem solutions might be hindered.

Investigations of Klauer (1992) suggest that reflecting activities must be co-ordinated with the problem solver's level of knowledge as reflective processes are not knowledge-generating processes. Knowledge can be decontextualised by reflection and thus, be transferred and applied to other situations. The function of reflection regarding cognitive activities is "... regulative, adaptive, integrative, organizing, anticipative and evaluative... " (Klauer 2001).

1. Questions and Hypotheses

In this study we want to investigate how useful the question-based reflection with own design solutions for experts is, i.e. students of mechanical engineering and professionally experienced technical designers. Therefore, the individual questions are:

(1) Does the QAT improve the solutions of advanced engineering students and of technical designers significantly?
Possibly the QAT might be helpful only for laymen as a systematic and complete proceeding might be already part of the education in engineering design and is thus being practised already before the intervention. Hence, we expect only small- to medium-sized effects as to the improvements after the QAT-intervention for these groups.

(2) Does the number of possible improvements of design solutions depend on the specific kind of the reflection?
For laymen we could not show any significant differences in the improvements between different modes of answering product specific, interrogative questions (Hacker and Wetzstein 2004; Wetzstein and Hacker 2004; Winkelmann et al. 2003).
In the industry mainly two versions of the QAT are of practical interest: answering non-product specific questions silently for oneself or answering them aloud to a questioner. Therefore, we will analyse whether these versions will show different effects in engineering designers with different levels of occupational experience. Corresponding with laymen we do not expect significant differences in the quality of solution after answering the interrogative questions silently for oneself vs aloud to a naive questioner.

(3) Do the possible improvements of the solutions provoked by the QAT depend on the experience of the designers on the job?

As non-product specific, interrogative questions offer no additional information, we assume that more experienced designers with their broader knowledge will profit more from these questions and thus, produce higher improvements than job beginners and students of mechanical engineering.

2. Methods

2.1. SAMPLE

42 students of engineering design of the Dresden University of Technology volunteered to take part in this study. The participants were all male, at an average age of 24 (SD = 2.2 years). Furthermore, 33 male engineering designers, at an average age of 35 years (SD = 7.8 years), who are employed in the automotive industry also participated.

2.2. TASK AND MATERIALS

The participants were asked to design a garden grill by manual sketching. This task was applied on the basis of investigations in co-operation with mechanical engineers.

The grill should meet the following requirements: (1) the distance between grid and coal pan should be fully adjustable between 5 and 25 cm; (2) for adjusting this distance grid and coal pan should not have to be touched directly; (3) grid and coal pan should be easily withdrawn for cleaning; (4) the equipment should not contain electrical components; (5) the grill should be stable. Dimension data were not required.

This task and the relevant evaluation scheme were developed by the Chair of Product Development at the Munich University of Technology. Corresponding with our interest in conceptual design, i.e. idea development, the participants should produce a manual paper sketch without the application of CAD.

The design processes were audio- and videotaped. For solving the design task the participants were given a sheet of paper, a pencil and a rubber. For any modifications during and after the intervention, the participants received a different coloured pen.

2.3. DESIGN, VARIABLES AND PROCEDURE

We applied a repeated measurement design (2x2x2) with the factors “type of intervention” (answering a list of non-product specific questions silently vs aloud to a questioner), “experience” (designers vs students of engineering design) and “time” (the measurement of solution quality before and after the intervention (pre-post-test), Table 5).

In this study we did not apply a control group without presenting the question list, Table 6, because in our former research only marginal differences between spontaneous reconsidering effects and systematic QAT-application could have been identified.

TABLE 5. Design.

Experience in engineering design (a)	Kind of intervention (b)			
	Answering non-product specific questions			
	silently for oneself		aloud to a questioner	
	Time of measurement (c)			
	Pre	*Post*	*Pre*	*Post*
Students	Y_{111}	Y_{112}	Y_{121}	Y_{122}
Designers	Y_{211}	Y_{212}	Y_{221}	Y_{222}

TABLE 6. Non-product-specific questions.

Devices
1. What are the devices of the object which had to be designed?
Requirements of the object
2. What are the explicit requirements for part A, i. e. expressed in the task?
3. Which requirements are implicit, e. g. arise indirectly from the task?
4. If you look at the total product, are there further requirements of the single devices? If so, which (ones)?
5. Which further demands result from each device's relation with other devices for itself?
6. If you put all the requirements of the devices together, which modifications and additions can be obtained then?
Explaining-Justifying-Evaluation
7. Please describe, how did you fulfil the requirements of the single devices?
8. Why did you decide on this variant? What are the advantages?
9. What are the disadvantages?
Improvement
10. What could be improved in your solution or what could a new solution look like? Why?
Interaction
11. If you did some modification on your solution, how can the devices still be combined?
12. If not, what do you have to do to put the devices together to a total product? Why?

The quality of the design before and after the intervention was evaluated by an independent engineering expert based on a developed evaluation scheme (Winkelmann 2005). It assessed the degree of accomplishment of 20

task-related requirements with a score ranging from 0 points (missing or not functioning) to 4 points (completely available or fully functioning), with a total maximum of 80 points.

The experiments were conducted separately for each participant. After finishing the design, participants were randomly assigned to one of the groups. The participants were instructed either to reconsider their solutions with a system of questions by answering these questions silently for oneself or aloud to a naive questioner. They were given the opportunity to revise their design with a different coloured pen during and after the intervention or to provide a completely new solution, Figure 1. The experiments lasted 1.5 hours on average. For all statistical tests (ANOVA, *U*-test and *t* test) we used an alpha level of .05.

3. Results

3.1. EFFECTS OF INTERROGATIVE QUESTIONS AND TYPE OF ANSWERING

Table 7 shows the mean quality of solution quality before and after the intervention and the mean improvement (difference). Before intervening, the intervention groups did not differ significantly ($F(3, 74) = 0.48$, $p > .05$).

TABLE 7. Means (M) and standard errors (SE) of the quality of solution before and after intervention and number of persons with improvement (%) (N = 75).

Kind of intervention	n	before intervention	after intervention	improvements (difference)	persons with improvements
		M ± SE	M ± SE	M ± SE	%
Students					
answering the questions silently	20	47,45 ± 1.78	48.05 ± 1.88	+ 0.60 ± 0.37	55
answering aloud to a questioner	22	48.41 ± 1.43	49.77 ± 1.42	+ 1.36 ± 0.30	59
Designers					
answering the questions silently	16	49,88 ± 1.43	51.06 ± 1.21	+ 1.19 ± 0.75	31
answering aloud to a questioner	17	49.53 ± 1.56	50.06 ± 1.58	+ 0.53 ± 0.21	29

The main effect "time" is significant ($F(1, 70) = 19.72$; $p < .05$, $\eta^2 = 0.22$). That means a significant improvement by the intervention. In contrast, the main effects "experience" and "type of intervention" do not show significant differences ($F_{experience}(1, 70) = 1.21$, $p > .05$; $F_{intervention}(1, 70) = 0.04$, $p > .05$).

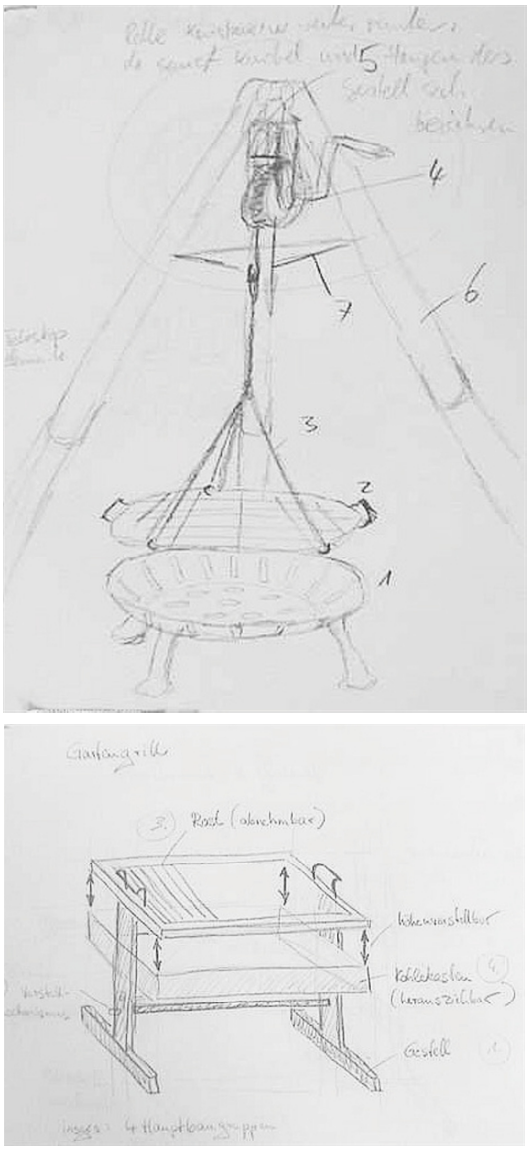


Figure 1. Design sketches.

Both are not of main interest to our problem as it integrates pre-test and post-test values and, thus, does not offer any information on the effects of the intervention. Rather, the interactions offer the essential messages: there is neither a significant interaction between the pre-post-test comparison and the "experience" (F (1, 70) = 0.01, $p > .05$) nor between the pre-post-test comparison and the "type of intervention" (F (1, 70) = 0.13, $p > .05$). This means that there is neither a difference between the numbers of improvements of the quality of solution between students and engineering

designers nor between the two types of interventions. However, there is a significant interaction of the three factors (F (1, 70) = 4.12, $p < .05$, η^2 = 0.06). Consequently, when the students answer the questions aloud to a questioner there are higher improvements in the solution quality than answering them silently for oneself (U-Test: z = - 2.14, $p < .05$, one-sided) and in comparison to the engineering designers (U-Test: $z_{silently}$ = -1.51, p = .86, one-sided; z_{aloud} = -1.81, p = .05, one-sided).

Hypothetically, the working time might be a covariate, explaining some results mentioned. However, we identified neither a significant correlation between the time spent on the sketches before the intervention and the quality of the solutions (r = 0.09, $p > .05$), nor between the length of the intervention period and the solutions (r = - 0.01, $p > .05$).

3.2. QUESTION-ANSWERING AND OCCUPATIONAL EXPERIENCE

The impact of the occupational experience of the engineering designers was analysed be dichotomizing this group into job beginners (up to 1.5 years) and advanced designers along with advanced designers for an ANOVA with a repeated factor. There is a significant main effect for the pre-post-test comparison (F (1, 30) = 9.95, $p < .05$, η^2 = 0.25), showing that both groups of designers improved their solutions by the interventions. By contrast, there is no significance of the main effect "experience" (F (1, 30) = 0.01, $p > .05$). However, there is a significant interaction of the pre-post-test comparison and experience (F (1, 30) = 5.30, $p < .05$, η^2 = 0.15) indicating that job beginners improved their solutions by the intervention more than advanced designers (U-Test: z = - 2.07, $p < .05$).

4. Discussion

The main question of this study was, whether a system of non-product specific interrogative questions may improve the design solutions of engineering designers, as this question had only analysed for laymen until then.

It is important to realise that the improvements by the QAT applied here are due to a system of non-product specific, interrogative questions which may be applied to any design task. Thus, this type of QAT is a generic tool.

This list of questions aims at the system of semantic relations as the questions, e.g. why asks for answers to describe causal, conditional, final etc. relationships. The description of these relationships will induce designers to carry out a more thorough analysis of the own results and – at least occasionally – to a further improvement, a modification or correction of some aspects of the solution obtained so far.

It may be supposed that professional designers are qualified to ask these questions spontaneously already before and without our intervention. Both, engineering students and engineering designers working in the German automotive industries improved their results significantly by the intervention

with the QAT. This holds, although most of the designers reported they are used to reconsider their solutions by answering themselves questions. However, these questions are specific content-related ones, but not interrogative questions aiming at the semantic relations of the object or system to be designed.

The improvements are higher for job beginners than for engineering designers with more experience.

The two different kinds of our interventions, answering the questions silently for oneself vs answering them aloud to a questioner did not differ significantly as to the number of the mean improvements.

Moreover, there is an essential exception as to the effects of the two kinds of intervention: the mean improvements of engineering design students are higher if they were asked by a questioner and had to answer aloud. Obviously, they profit from the necessity to reconsider their solutions more completely and systematically than if answering only silently for oneself.

All in all, the non-product specific QAT turns out to be useful for both professional engineering designers – especially for job beginners – as well as for engineering design students.

The generic tool is applicable for several design tasks. At least for laymen we did not find significant differences in the improvements for different design tasks (Winkelmann 2005). Nevertheless, further research concerning this matter is necessary.

However, there are also clear limitations of the QAT: the improvements obtained so far only cover about 5% of the possible range for improvements and only 30 up to 50% of the participants improved their solutions by the intervention in spite of their large possibilities for an improvement. The main reasons of these limitations are, at first, that a system of questions does not offer any additional information that might be applied in the design procedure. Secondly, asking questions by a naive questioner does not mean a dialogue with an exchange of arguments and the offer of alternative ideas.

Consequently, higher improvements should be possible by an extension of the QAT beyond the laboratory-like research scenario which offers the possibility to look for additional information and to discuss solutions and question-based alternatives with professional colleagues, i.e. to enable a dialogue in form of a discourse.

References

Adams, RS, Turns, J and Atman, CJ: 2003, Educating effective engineering designers: the role of reflective practice, *Design Studies* **24**: 275-294.

Ahmed, S, Wallace, KM and Blessing, LTM: 2000, *Training Document – C-QuARK Method – The Experienced Designer's Approach to Design*, Unpublished, Engineering Design Centre, Cambridge University, Cambridge.

Akin, Ö: 1978, How do architects design?, *in* JC Latombe (ed), *Artificial Intelligence and Pattern Recognition in Computer-Aided Design*, New York, North Holland, pp. 65-104.

Atman, CJ and Turns, J: 2001, Studying engineering design learning: Four verbal protocols studies, *in* CM Eastman, WM McCracken and WC Newstetter (eds), *Design Knowing and Learning: Cognition in Design Education*, Elsevier, Oxford.

Bartl, C and Doerner, D: 1988, Sprachlos beim Denken – zum Einfluss von Sprache auf die Problemlöse- und Gedächtnisleistung bei der Bearbeitung eines nicht-sprachlichen Problems, *Sprache & Kognition* **17**(4): 224-238.

Birch, H: 1975, The relation of previous experience in insightful problem solving, *Journal of Comparative Psychology* **38**: 367-383.

Cottrell, S: 2003, *Skills for Success, The Personal Development Planning Handbook*, Palgrave Macmillan, Houndmills.

Cross, N: 2001, Design cognition: Results from protocol and other empirical studies of design activity, *in* CM Eastman, WM McCracken and WC Newstetter (eds), *Design Knowing and Learning: Cognition in Design Education*, Elsevier, Oxford.

Daudelin, M: 1996, Learning from experience through reflection, *Organizational Dynamics* **24**: 36-48.

Dominowski, RL: 1990, Problem solving and metacognition, *in* KJ Gilhooly, MTG Keane, RH Logie and G Erdos (eds), *Lines of Thinking: Reflections on the Psychology of Thought*, Wiley, Chichester, England, Volume 2, pp. 313-328.

Doerner, D: 1998, *Mannheimer Beiträge zur Wirtschafts- und Organisationspsychologie*. Sonderheft, Zukunft der Kognitionspsychologie, Kolloquium am 21.11.1997 anlässlich der Verabschiedung von Prof. Dr. Theo Hormann. Universität Mannheim, Lehrstuhl für Wirtschafts- und Organisationspsychologie.

Eastman, CM: 1969, Cognitive processes and ill-defined problems: A case study from design, *Proc. of the First Joint International Conference on Artificial Intelligence*, Washington, DC, pp. 675-699.

Ericsson, KA and Simon, HA: 1993, *Protocol Analysis: Verbal Reports as Data* (revised edition), MIT Press, Cambridge, MA.

Eris, O: 2003, How engineering designers obtain information, *in* U Lindemann (ed), *Human Behaviour in Design*, Springer, Berlin, pp. 142-153.

Ertmer, P and Newby, T: 1996, The expert learner: Strategic, self-regulated and reflective, *Instructional Science* **24**: 1-24.

Goeker, MH: 1997, The effects of experience during design problem solving, *Design Studies* **18**(4): 405-426.

Hacker, W and Wetzstein, A: 2004, Verbalisierende reflexion und lösungsgüte beim Entwurfsdenken. *Zeitschrift für Psychologie* **212**(3): 152-166.

Halton, JH and Smith, GB: 1995, Radical-inverse quasi-random point sequence, *Communications of the ACM* **7**(12): 701–702.

Holyoak, KJ: 1991, Symbolic connectionism: Toward third-generation theories of expertise, *in* KA Simon and J Smith (eds), *Toward a General Theory of Expertise: Prospects and Limits*, University Press, Cambridge.

Klauer, KJ (ed): 2001, *Handbuch Kognitives Training*, Hogrefe-Verlag, Göttingen.

Kluge, A: 1999, *Erfahrungsmanagement in Lernenden Organisationen*, Verlag für Angewandte Psychologie, Göttingen.

Pahl, G and Beitz W: 1997, *Konstruktionslehre*, Springer-Verlag, Berlin.

Rhone, E, Turns, J, Atman, CJ, Adams, R, Chen, Y and Bogusch, L: 2001, *Analysis of Senior Follow-up Data: The Midwest Floods Problem-Addressing Redundancies in Coding*, Center for engineering Learning and Reaching (CELT). Technical Report #01-05, University of Washington, Seattle.

Strzalka, J and Strzalka, F-J: 1986, Perspektivisches denken und reflexionen beim lösen eines komplexen problems, *Sprache und Kognition* **4**: 202-210.

Wallace, K and Ahmed, S: 2003, How engineering designers obtain information, *in* U Lindemann (ed), *Human Behaviour in Design,* Berlin: Springer, pp. 184-194.

Wetzstein, A: 2004, *Unterstützung der Innovationsentwicklung. Einfluss von wissensbezogenen Interaktionen insbesondere im Kooperativen Problemlösen und Fragenbasierter Reflexion*, Theorie und Forschung, S. Roderer Verlag, Regensburg.

Winkelmann, C: 2005, Die Fragetechnik für den Konstrukteur: *Eine fragenbasierte Unterstützung der frühen Phasen des konstruktiven Entwurfsprozesses*, Roderer Verlag, Regensburg.

Winkelmann, C: 2006, Design problem solving: Wovon hängen lösungsgüteverbesserungen durch eine frage-antwort-technik ab?, *Zeitschrift für Psychologie* 1.

Winkelmann, C, Wetzstein, A and Hacker, W: 2003, Question Answering – vergleichende bewertung von reflexionsanregungen bei entwurfstätigkeiten, *Wirtschaftspsychologie* **1**: 37-40.

CONTEXTUAL CUEING AND VERBAL STIMULI IN DESIGN IDEA GENERATION

LASSI A LIIKKANEN
University of Helsinki, Finland

and

MATTI K PERTTULA
Helsinki University of Technology, Finland

Abstract. This paper presents an initial empirical test of a cognitive model of memory search in idea generation. In the experiment, we examined how manipulations in contexts and cues affect the structure of subsequently generated ideas. The study shows that these manipulations change the categorical frequencies of generated ideas. The results are generally inline with central assumptions of the model.

1. Introduction

The design of a new product begins with conceptual design activities, which have a decisive influence on the properties of the final design. Conceptual design includes an internal search stage, in which a designer explores different alternatives for the final design. This part is also called idea generation. An effort called design cognition using the vocabulary of cognitive sciences has recently been applied to design research. Several different approaches within this paradigm have been taken to investigate various parts of the design process, including idea generation.

To describe design idea generation, we have developed a model called *Cue-based Memory Probing in Idea Generation* (CuPRIG), intended to describe memory search within the idea generation process (Perttula and Liikkanen, 2005). The model proposes that memory search is a cue and context dependent process. In this paper, we present an initial empirical test of CuPRIG, in order to demonstrate how manipulations in context and cue affect the contents of subsequently generated ideas.

J.S. Gero (ed.), Design Computing and Cognition '06, 619–631.
© 2006 *Springer. Printed in the Netherlands.*

1.1. OUTLINE OF CUPRIG

The CuPRIG model follows a knowledge-based design approach by assuming that idea generation is a cognitive activity that relies heavily on previously acquired knowledge (Ward 1994). In this view, idea generation performance is determined by one's ability to efficiently access and retrieve information stored in a database, which corresponds to long-term memory (LTM) in humans.

CuPRIG attempts to model the internal search phase of the idea generation process by the concept of memory probing, which refers to activating items (knowledge) in long-term memory. CuPRIG adapts several concepts from the associative memory search model presented by Raaijmakers and Shiffrin (1981). We outline the idea generation process to include the following main phases:

- *Interpretation:* The subject interprets the problem statement, and any other stimulus materials available to form an internal representation of the problem.
- *Retrieval*: The subject assembles a search cue in his working memory (WM), which is used together with a context to probe LTM in order to retain task-relevant knowledge.
- *Adaptation:* The retrieved knowledge structures are synthesized to meet the task constraints of the current situation using additional cognitive processes.

Item retrieval from LTM to WM is assumed to be a cue and context dependent process, which is governed by some function of similarity between the semantic elements of the probe and LTM items. The process is probabilistic rather than deterministic; given a cue and a context (semantic properties) we can only estimate the most probable outcome rather than to be certain of what be retrieved. The items of LTM are considered to be semantic knowledge, which could be simple items with functional properties or more complete models, such as schemata (Gero 1990). CuPRIG does not assume any particular type of design knowledge representation, therefore this issue is not considered here in greater depth.

1.2. MODEL TESTING

To test the model, we will evaluate whether changes to the cue and/or the context affect the structure of following ideas in a systematic way. For this purpose, the cue and context must be defined at a concrete level.

We assume that the cue has semantic properties normally attributed to a verb in natural language; it refers to some action or function. It should be noted that the phrases used as examples here exist on the surface level of language, where as the cue and context that CuPRIG refers to are located on a deeper, semantic level (Chomsky 1957). When considering design

problems, a verb could be an abstract basic function, such as “to transfer”, which can apply to almost anything, or a more detailed word such as “to pump”, which is usually related to liquids. The cue is likely to be complemented with a noun that determines object, for instance, “oil”. We assume that the verb and the noun make up crucial parts of the (semantic) cue used in memory retrieval, with an emphasis on the former. It is easy to understand that the design of a machine that *pumps* oil is very different from that of a machine that *burns* oil. If we omit one or the other from the linguistic description, then it will become impossible to determine even a considerably abstract machine. The role of the cue is therefore critical. However, this also means that it is impossible to modify the cue within a particular problem description without turning the task into something else. Therefore, we will not manipulate the cue directly. Instead, we will use a single word as a stimulus cue for the design task and we will investigate how simple verbal stimuli can activate LTM items and how these items are fused to the actual design in the adaptation process. In other words, the idea is to imply a verb (i.e. function) through cueing with a noun (i.e. object). We hypothesize that the items activated by the stimuli will be synthesized to new ideas in a coherent way as has been observed in the studies of analogical mapping (Hummel and Holyoak 1997) and conceptual synthesis (Costello and Keane 2000).

Context is commonly accepted to affect both information encoding and retrieval, although its definition varies in the psychological literature (Eysenck and Keane 2001). Contextual memory can be further subdivided into interactive and independent context (Baddeley 1982). Independent (or extrinsic) context refers to adding a temporal or a spatial property to an item upon encoding, e.g. things learned on the first grade or learned in the swimming pool. Interactive (or intrinsic) context, in turn, is defined as a semantic feature of a stored item; say pumping oil on a platform and pumping oil on ground is an example of changing the context of a function. Hence, neither context type changes the function of the item itself. However, knowledge of contextual attributes of items can be used e.g. to improve recall through contextual cueing. In the current study, unlike in the Search of Associative Memory (Raaijmakers and Shiffrin 1981), which discussed the role of independent context, we study alterations in the interactive context, with the hypothesis that these manipulations will significantly affect the probability of retaining particular items.

In short, we believe that indirectly imposed changes in cues and manipulation of context will produce regular variation in the structure of ideas in a way that can be understood in terms of the CuPRIG model.

2. Methods

2.1. SUBJECTS

Fifty students of mechanical engineering at the Helsinki University of Technology participated in the experiment. The students were predominantly male (93%). Their average age was 23 (SD = 2) years and only 6% had more than one year experience from practical design. The mean curriculum phase was 98 (SD = 24) study credits completed from a total of 180 required for a Master's degree. Therefore, the subjects should be considered as novice designers.

2.2. EXPERIMENT DESIGN AND PROCEDURE

The experiment was designed to evaluate whether changes in contexts and cues affected the structure of generated ideas in a systematic way. Subjects were asked to generate a design idea after being forced an external cue or a particular interactive context. They were tested simultaneously while seated in auditorium type classroom. The experiment included two tasks; Ball and Plant, Table 1.

TABLE 1. Task descriptions for the two different design problems used in the experiment.

NAME	TASK ASSIGNMENT
Ball	An automatic device that transfers balls from a playing field to a goal-area.
Plant	An automatic watering device for houseplants. The device should provide a plant with about a deciliter of water per week - no more, no less. It should be able to water the plant for one month.

A cue and context manipulation was induced for both tasks. The manipulations were administered by written statements on the answering sheet. The cue manipulation included a simple keyword that was to be used to 'awaken thoughts'. The keyword was 'water' for the Ball task and 'sun' for the Plant task. The keyword was given to half of the participants; the other half was a control group for this manipulation and received no external stimulus. Context manipulations were administered by using written clarifications of the (independent) context. The context descriptions are shown in Table 2.

Subjects were asked to generate four ideas for both of the tasks. The context manipulation was presented before the first idea, and the keyword before the third idea.

TABLE 2. Contextual descriptions used for the context manipulations for the two design problems.

TASK	CONTEXT DESCRIPTIONS
Ball	Balls are located in a soccer field;
	Balls are located on a board that is the size of a dining-table.
Plant	The plant is located in a living room;
	The plant is located in a greenhouse.

2.3. DATA ANALYSIS

The objective of the experiment was to examine whether the context manipulation and the presentation of a keyword led to different responses between the experimental conditions. We also included a control group, in which no context or cue manipulation was induced. The control group data was acquired from results of prior experiments that used identical task descriptions. The control data included responses from twenty-four persons for the Plant task and from ninety persons for the Ball task.

The metric chosen for this assessment was categorical frequency. *Categorical frequency* describes the distribution of ideas into pre-defined solution categories. A solution category is a cluster of solutions that are genealogically linked to each other (e.g. Nijstad et al. 2002). Fisher's exact tests were used to evaluate the statistical significance of differences in the categorical frequencies between experimental conditions. Although subjects produced two ideas in response to a manipulation, the categorical frequency assessment was performed on the first idea generated after a manipulation.

There are several possible approaches to classifying ideas into general solution categories. Individual solutions can be classified as such, based on the primary means of satisfying the main function. Alternatively, the classification can be based on a functional decomposition scheme (Pahl and Beitz 1984), which is a suggested approach for assessing solutions for engineering problems (Shah et. al. 2003). Functional decomposition refers to decomposing the main function into its primary sub-functions. In this experiment, we applied both tactics.

The solutions for the Ball task were classified into a single category. Classification based on sub-functions was not used because the solutions were rather simple and could easily be classified into a single category. In turn, a decomposition-based classification was used for categorizing the solutions for the Plant task. Four primary sub-functions were identified: water source, regulation, water transfer, and energy source. These sub-functions were assumed to present separate and meaningful parts of the design.

3. Results

3.1. CONTEXTUAL CUEING

3.1.1. The Ball task

The categorical frequencies in the two different contexts and in the reference condition are shown in Table 3. The statistical tests showed that the difference in categorical frequency between the soccer-field and board context was statistically significant (Fisher's exact test, $p = .021$). The categorical frequencies differed also significantly between the board-context and the reference condition (Fisher's exact test, $p = .001$), but not between the soccer-field context and the reference condition (Fisher's exact test, $p = .367$).

TABLE 3. Categorical frequency of generated ideas in two different contexts for the Ball task. Reference data (no context manipulation) is also shown in the table.

CLASS	CONDITION		
	Reference (N = 90)	Soccer-field (N = 25)	Board (N = 23)
Free-moving collector	52	14	5
Leveler	10	3	5
Conveyor	6	3	1
Thrower	8	3	1
Inclinator	7	1	6
Fixed collector	0	1	3
Other*	7	0	3

*) Number of ideas from categories that occurred in less than 10 percent of cases in each condition.

3.1.2. The Plant task

The categorical frequencies for the three different conditions are shown in Table 4. The statistical tests did not show significant differences between the conditions (Fisher's exact tests, $p > .100$).

3.2. VERBAL STIMULUS

3.2.1. The Ball task

The keyword 'water' was presented to half of the participants in the two different contexts after subjects had completed two ideas. Categorical frequencies in the different conditions are shown in Table 5. The statistical tests showed that the categorical frequencies were different between the

stimulus and control condition in the soccer-field (Fisher's exact test, p = .007) and the board context (Fisher's exact test, p = .028).

TABLE 4. Categorical frequency of generated ideas in two different contexts for the Plant task. Reference data (no context manipulation) is also shown in the table.

SUB-FUNCTION	CONTEXT		
CLASS	Reference (N = 24)	Living-room (N = 25)	Greenhouse (N = 24)
WATER SOURCE			
Separate tank	18	16	18
Water-pipe	2	2	3
Integrated Tank	2	4	0
Other*	0	1	0
Not defined	1	2	3
REGULATION			
Timer	6	8	14
Steady flow	10	8	6
Mould humidity	3	0	0
Other*	0	2	0
Not defined	5	7	4
WATER TRANSFER			
Drained	12	15	15
Released	0	3	0
Pumped	3	2	3
Absorbed	4	2	0
Other*	5	1	3
Not defined	0	2	3
POWER SOURCE			
Unnecessary	14	17	16
Other*	3	2	0
Not defined	7	6	8

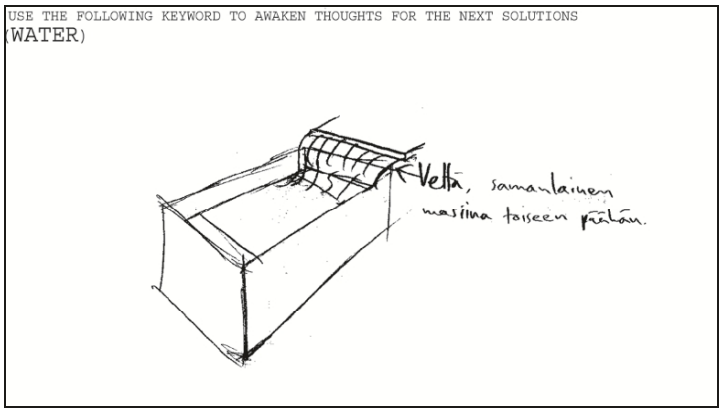
*) Number of ideas from categories that occurred in less than 10 percent of cases in each condition.

Figure 1 presents a pair of typical ideas generated after the stimulus; 67% of the total ideas (N = 21) generated after presenting the keyword 'water' were from the two categories (cannon and laminar flow) presented in the figure.

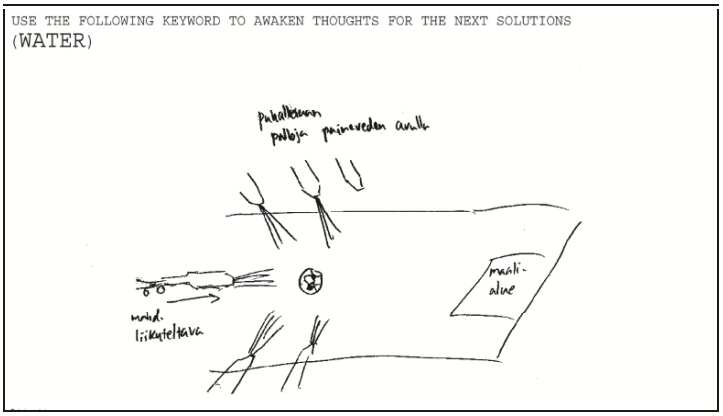
3.2.2. The Plant task

The keyword 'sun' was presented to half of the participants before the third idea was generated. Categorical frequencies in the different conditions are shown in Table 5. The statistical tests showed that the categorical frequency was different for sub-function 'regulation' in the living room (Fisher's exact test, p = .002), but not in the greenhouse context (Fisher's exact test, p = .159) contexts. Similarly, a statistically significant difference occurred

for sub-function ‘water transfer’ in the living room context (Fisher’s exact test, p =.008), but not in the greenhouse context (Fisher’s exact test, p = .352). The differences were statistically significant for sub-function ‘power source’ in both of the contexts (living room: p=.000, greenhouse: p =.001). However, there were no significant differences in the categorical frequencies for sub-function ‘water source’ (Living room, p = .066; Greenhouse, p = 1.00).



(a)



(b)

Figure 1. Typical ideas from generated after cueing with the stimulus word ‘Water’; (a) laminar flow, (b) Cannon.

Figure 2 presents characteristic ideas generated after presenting the stimulus word ‘sun’. The stimulus did not change the idea completely; instead the stimulus had a clear effect on some of the sub-function solutions. One or more of the aspects shown in the figure was changed in 95% (N = 20) of ideas generated after cueing.

TABLE 5. Categorical frequencies for ideas generated after a verbal stimulus in the Ball task. Reference data from a non-cued condition is also shown in the table.

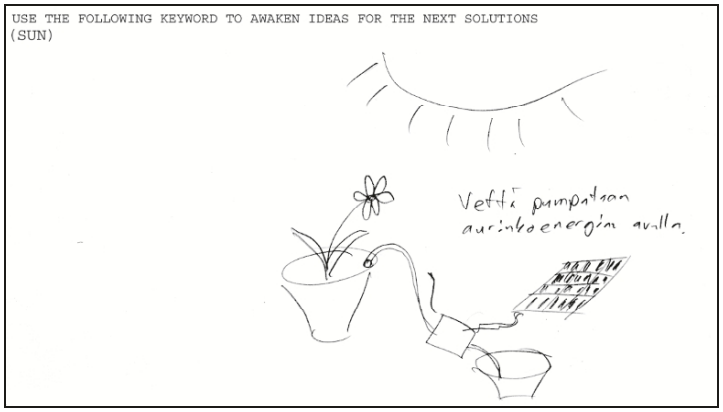
	Soccer-field		Board	
	Control (N = 11)	Stimulus (N = 12)	Control (N = 13)	Stimulus (N = 9)
Free-moving collector	5	3	3	1
Leveler	2	0	1	0
Inclinator	1	1	2	1
Water-cannon	0	7	0	3
Water-flow	0	1	0	3
Conveyor	0	0	1	1
Other*	3	0	4	0

*) Number of ideas from categories that occurred in less than 10 percent of cases in each condition.

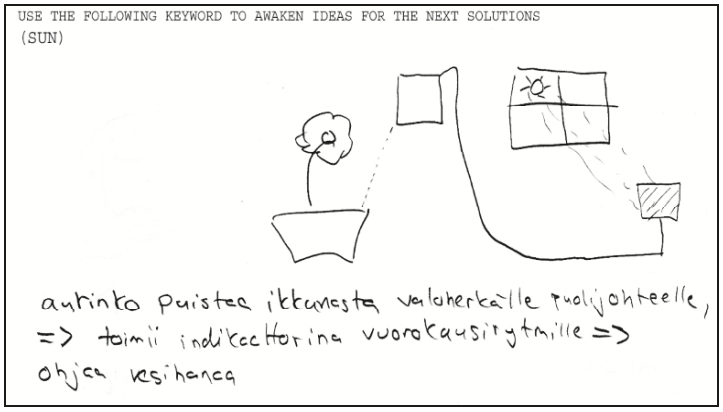
4. Discussion

Idea generation is one of the most important stages in the design process. Understanding idea generation is required for evaluating and developing methods and tools to support this activity (e.g. Shah et. al 2000). The current view is that idea generation should be understood as a memory-based activity, which is sensitive to external stimuli and internal constraints (Smith 1995). To this end, we have developed a cognitive model of memory search in idea generation called Cue-based Memory Probing in Idea Generation (CuPRIG). CuPRIG treats idea generation as a cue and context dependent internal search process. In this paper, we presented an empirical test of the model, which was designed to assess its primary elements. We evaluated how manipulations of contexts and cues affected the structure of subsequently generated ideas. More precisely, we compared categorical frequency distributions of generated ideas after inducing manipulations of contexts and cues. Two different tasks (named Ball and Plant, see Table 2 for descriptions) were used in order to avoid bias towards a single task. The results of the empirical tests are generally inline with the central assumptions of the model.

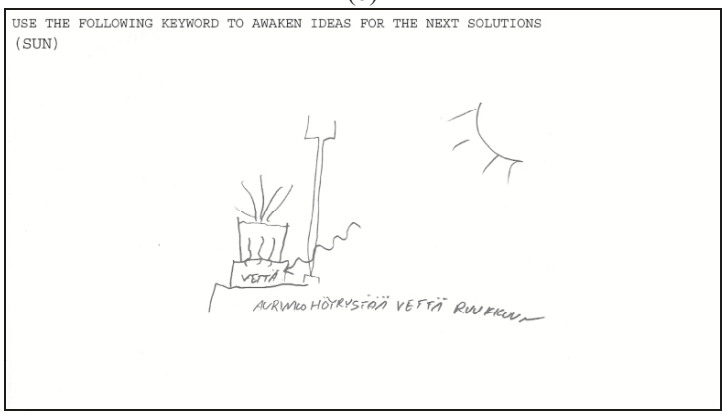
The first element tested was the role that interactive context has in determining which items are sampled from memory. The idea was that since a context is intrinsically encoded to an item, this information should facilitate its recovery. In the experiment, subjects were induced to particular contexts prior to generating ideas. Contexts were descriptions of the environment or surroundings in which the future design was to operate. In the Ball task, there was a clear effect for contextual cueing; probabilities of generating ideas from particular categories changed in regards to the context in which the subjects imagined the design.



(a)



(b)



(c)

Figure 2. Typical ideas generated after cueing with the keyword ‘Sun’; (a) ‘Solar-energy’ (sub-function: Power source), (b) ‘Sun-light’ (sub-function: Regulation), (c) ‘Vaporized’ (sub-function: Water transfer).

However, no difference was found between the control condition and one of the contexts (soccer-field) in the Ball task. The reason may be that the soccer-field (or equivalent) is a common context and implicitly assumed by the majority of subjects from the task description, whereas board-like contexts are more unique. Furthermore, no difference between contexts was found in the Plant task. There are a few possible explanations for this difference. One explanation is that the different approaches used to classify the ideas into categories caused the disparity between the two tasks. On the other hand, it would be expected that the categorization based on functional decomposition (used for the Plant task) should be more sensitive, because it provides a more fine-grained analysis of the content of the ideas. A second, more probable, reason for this inconsistency could be that the selected contexts for the Plant task were semantically too close to each other in order for them to cause sampling of different items.

TABLE 6. Categorical frequencies for ideas generated after a verbal stimulus in the Plant task. Reference data from a non-cued condition is also shown in the table.

	Living room		Greenhouse	
	Control (N = 10)	Stimulus (N = 11)	Control (N = 10)	Stimulus (N = 9)
WATER SOURCE				
Separate tank	5	10	6	7
Water-pipe	3	0	1	1
Other*	1	1	1	0
Not defined	1	0	2	1
REGULATION				
Timer	4	1	3	4
Steady flow	1	0	2	0
Sun-light	0	8	0	2
Other*	4	1	3	0
Not defined	1	1	2	3
WATER TRANSFER				
Drained	6	3	4	2
Released	0	3	2	2
Pumped	0	2	0	2
Vaporized	0	3	1	1
Other*	4	0	1	2
Not defined	0	0	2	0
POWER SOURCE				
Unnecessary	1	0	2	0
Solar energy	0	9	0	7
Not defined	9	2	8	2

*) Number of ideas from categories that occurred in less than 10 percent of cases in each condition.

The second element that was tested was the retrieval cue itself. The cue should be understood as an abstract action or function, whereas cueing refers to attempting to match that function to memory items that afford the required functionality. We did not manipulate the cues directly; instead we used keywords representing nouns in natural language. The idea was that the keyword would imply a function that could satisfy the design requirement. Altering the cue had a clear impact in both tasks. The keyword water was used as stimulus in the Ball task. The majority of the subjects generated designs in which water was used as the mediating substance to move the balls, the actions themselves were either to cause a laminar flow to move the balls or to shoot the balls with cannon-like devices. In the Plant task, the effect was more focused i.e. biased to certain parts of the design. A significant impact occurred for sub-function 'power source' in both contexts, whereas the cue influenced sub-functions 'regulation' and 'water transfer' only in the living room context. The most frequent idea that the stimulus evoked was the use of a solar-panel to capture the energy in sun-light to operate the device. Other ideas associated with the keyword were to regulate the device with sun-light or to passively vaporize water for the plant. This impact could have been determined by the associative strength between the stimulus and associated items. Hence, solar power may have been more strongly associated with the keyword in the current situation than the two other related ideas. Moreover, this finding implicates that subjects may orientate their search efforts towards a single sub-function. Indeed, some designers actually stated that "this is idea is like the previous one, but gets energy from the sun".

Since there was discontinuity in the results, some further factors should be discussed. As the effect of the cue and context are dependent on one's knowledge structures, some randomness was expected in the results. Cues may simply activate different knowledge structures between individuals, resulting in the production of different ideas. This may be especially critical when comparing small groups such as in the verbal stimulus manipulation. Thus, it should be stressed that cueing possesses a probabilistic character. Based on this implication, we would anticipate differences to be dependent generally on one's disciplinary backgrounds and expertise. This notion does not however overrule the theory that contextual fluctuations and verbal stimulus affect idea production. Even that, cueing may affect subjects differently; it seems to change subjects' idea generation systematically. Therefore, the results are generally consistent with what could be estimated by the CuPRIG model. The cue leads to the activation of some semantic unit, which is then synthesized in a very straightforward manner to create a new idea. The new ideas seem to be generated in a very similar fashion over

the subjects, neglecting the variation in solution details, which is predicted by the model.

What does actual design practice have to gain from an experiment such as described here? The current study showed how minor changes in the task assignment can bring changes to the resultant ideas. Also, these differences can be explained by a rather systematic sampling of memory, governed by identifiable elements and cognitive processes. This type of knowledge and process-understanding can be used to improve procedures and techniques of idea generation and to avoid common pitfalls, such as, implicitly assuming contexts that are less favorable than others given the need for creative design outcomes.

References

Baddeley, A: 1982, Domains of recollection, *Psychological Review* **83**(6): 708-729.

Chomsky, N: 1957, *Syntactic Structures*, The Hague, Mouton.

Costello, FJ and Keane, MT: 2000, Efficient creativity: Constraint-guided conceptual combination, *Cognitive Science* **24**(2): 299-349.

Eysenck, MW and Keane, MT: 2000, *Cognitive Psychology. A Student's Handbook, (4th ed)*, Psychology Press, East Sussex.

Gero, JS: 1990, Design prototypes: A knowledge representation schema for design, *AI Magazine* **11**(4): 26-36.

Hummel, JE and Holyoak, KJ: 1997, Distributed representations of structure: A theory of analogical access and mapping, *Psychological Review* **104**(3): 427-466.

Jansson, DG and Smith, SM: 1991, Design fixation, *Design Studies* **12**(1): 3-11.

Nijstad, BA: 2000, *How the Group Affects the Mind,* Utrecht University.

Nijstad, BA, Stroebe, W and Lodewijkx, HFM: 2002, Cognitive stimulation and interference in groups: Exposure effect in an idea generation task, *Journal of Experimental Social Psychology* **38**(6): 535-544.

Pahl, G and Beitz, W: 1984, *Engineering Design*, The Design Council, London.

Perttula, M and Liikkanen, LA: 2005, Cue-based memory probing in idea generation, *in* JS Gero and ML Maher (eds), *Sixth Roundtable Conference on Computational and Cognitive Models of Creativity,* Key Centre of Design Computing and Cognition, University of Sydney, Sydney, pp. 195-210.

Raaijmakers, JG and Shiffrin, RM: 1981, Search of associative memory, *Psychological Review* **88**(2): 93-134.

Shah, JJ, Kulkarni SV and Vargas-Hernandez, N: 2000, Evaluation of idea generation methods for conceptual design: Effectiveness metrics and design of experiments, *Journal of Mechanical Engineering* **122**(4): 377-384.

Shah, JJ, Vargas-Hernandez, N, and Smith, SM: 2003, Metrics for measuring ideation effectiveness, *Design Studies* **24**(2): 111-134.

Smith, SM: 1995, Getting into and out of mental ruts: A theory of fixation, incubation, and insight, *in* RJ Sternberg (ed), *The Nature of Insight*, MIT Press, Cambridge, MA, pp. 229-251.

Ulrich, KT and Eppinger, SD: 2003, *Product Design and Development (3rd ed)*, McGraw-Hill, Boston.

Ward, TB: 1994, Structured imagination: The role of category structure in exemplar generation, *Cognitive Psychology* **27**(1): 1-40.

DESIGN COLLABORATION

Communicating, integrating and optimising multidisciplinary design narratives
John Haymaker

Enhanced design checking involving constraints, collaboration, and assumptions
Janet Burge, Valerie Cross, James Kiper, Pedrito Maynard-Zhang and Stephan Cornford

Collaborative virtual environments on design behaviour engineer at sketch time
Rodrigo Mora, Roland Juchmes, Hugues Rivard and Pierre Leclercq

DesignWorld: A multidisciplinary collaborative design environment using agents
Michael Rosenman, Kathryn Merrick, Mary Lou Maher and David Marchant

COMMUNICATING, INTEGRATING AND IMPROVING MULTIDISCIPLINARY DESIGN NARRATIVES

JOHN HAYMAKER
Stanford University, USA

Abstract. AEC professionals commonly use discipline-specific computer-based information modeling and analysis processes today. However, these professionals lack simple, flexible, formal frameworks to communicate and integrate these processes and information amongst multiple disciplines. They therefore struggle to quickly and accurately achieve balanced and near-optimal multidisciplinary designs. Narratives are formal, visual descriptions of the design process that include representations, reasoning, and their interrelationships. This paper presents several conceptual and implemented Narratives, and discusses how they can help AEC professionals better communicate and integrate their design processes and information and thus potentially improve their designs.

1. Introduction: Existing Methods Do Not Adequately Support the Narrative Nature of AEC

Designing and constructing successful buildings is becoming increasing complex. Projects must achieve an increasing number of economy, ecology, and equity goals, and must therefore involve multidisciplinary design and analysis (MDA). MDA demands that Architecture, Engineering, and Construction (AEC) professionals understand the interdependencies and make tradeoffs between their discipline-specific goals and the goals of other disciplines. These professionals must work under severe time and budget constraints; late, over-budget, and functionally unsatisfactory projects are common.

To achieve their goals, AEC professionals produce tremendous amounts of information describing everything from existing conditions to project goals and requirements, design options, design analyses, construction documentation, fabrication and installation information, as-built, operation and demolition information. When constructing this information, they often consult other information produced by other professionals and in other

J.S. Gero (ed.), Design Computing and Cognition '06, 635–653.
© 2006 *Springer. Printed in the Netherlands.*

project phases, disciplines, or industries. They need to maintain the integrity of their information as the project evolves.

In other words, AEC professionals develop what one might call *narratives* for their own work and interweave them with narratives of other engineers. The Oxford English Dictionary defines a *narrative* as "An account of a series of events, facts, etc., …with the establishing of connections between them." In AEC practice, narratives help professionals expose cross-disciplinary impacts and integrate their work with the work of other project stakeholders. However, today these narratives are often not adequately represented or managed. AEC professionals usually represent their information in either manual representations or computer files, but the connections between the information, in this case the *dependencies*, are not formally represented but rather stored in the heads of the professionals. This way of constructing, communicating, and managing multidisciplinary project information and processes is proving to be time-consuming, error-prone, and expensive.

The AEC industry could benefit from theory and methods that enable AEC professionals to more easily yet formally construct and manage their narratives to suit a project's unique social, cultural, economic, technical, environmental, and other criteria. Building on and extending prior work on representing, exchanging, and using engineering information and on frameworks to organize and manage the engineering design process, we at the Center for Integrated Facility Engineering are designing and implementing a generic language and framework for constructing and controlling formal Narratives consisting of information representations, and explicit dependencies between these representations.

This paper briefly summarizes prior observations about the multidisciplinary, constructive, iterative, and unique character of AEC projects, and the difficulty AEC professionals have communicating, integrating and optimizing their multidisciplinary processes and information on these projects. Next, the paper reviews the formalization of Narratives that we are designing to address this need, and discusses the AEC profession and related research with respect to representing and interrelating multidisciplinary project information. The paper then presents several conceptual and implemented Narratives and discusses the Narrative methodology's ability to enable AEC professionals and students to more quickly and accurately communicate and integrate their multidisciplinary design processes. Finally, the paper speculates on the ability of Narratives to help these professionals improve their designs, and discusses future work towards this goal.

2. Summary of Case Studies Illustrating the Narrative Structure of AEC Projects

This section briefly reviews two case studies that illustrated the implicit narrative structure of AEC projects. Both projects were internationally recognized for their state-of–the-art use of technology; however, the cases illustrate that due to a lack of formal support for the narrative nature of the design process, the professionals on these projects struggled to communicate and integrate their design processes and information, and thus to improve and balance their designs.

2.1. COST BENEFIT ANALYSIS FOR AN ATRIUM IN AN OFFICE BUILDING

In Haymaker et al. (2006), we describe and diagram cases that detail the design process a project team including architects, mechanical engineers, construction managers and other consultants performed to determine the costs and benefits of employing different design strategies in the schematic design phase of a headquarters office building. In one example, the team wanted to know what the costs and benefits of employing skylights and an atrium would be. They studied industry data that measured the improved productivity and reliability of the workforce in similar environments, and constructed a reasonable estimate for the expected productivity gain and absenteeism improvement in a strategically day lit space compared to a more traditional, artificially illuminated space. As a business, they needed to weigh this expected productivity gain against the expected lifecycle cost of constructing and operating the building. To calculate this cost, they asked what the added construction cost and potential energy savings (due to the reduction in artificial light) would be. In order to answer these questions, they needed to ask how much natural light would enter the building should different combinations of skylights and atria be employed. In order to answer these questions, they needed to ask what a building with and without atria and skylight might look like. In order to answer these questions, they asked about the client's requirements, the prevailing regulatory requirements, and the characteristics of the site. An implicit narrative of interrelated design and analysis representations were constructed to explore and answer these questions. While the representations were explicit, the dependencies between these representations were not formally described or managed.

2.2. DESIGN AND FABRICATION OF DECK ATTACHMENTS FOR A CONCERT HALL

In Haymaker et al. (2004), we describe how a design and construction team detailed and fabricated the structural system for a concert hall during the construction phase. The architect constructed and integrated a representation

describing the boundary of each concrete slab on the project. From this and other representations, the structural engineer constructed a representation describing the centerline of each steel member required for the frame of the building. Using this information, the steel detailer constructed a representation describing the boundary of each steel member and its fasteners. The metal decking detailer then constructed a representation describing where to install deck attachments to connect the metal decking for concrete floor slabs to the structural beams. The steel fabricator fabricated the beams, welding the required deck attachments to the respective beams in the shop. Again, while these representations were explicit, the dependencies between these representations were kept and managed only in the heads of these professionals.

2.3. OBSERVATION: AEC PROFESSIONALS STRUGGLE TO COMMUNICATE, INTEGRATE THIER NARRATIVES, AND THUS BALANCE AND OPTIMIZE THEIR DESIGNS

In the cases, we observed that AEC professionals had difficulty:

- *Communicating these design processes and information:* No diagram or other formal description of these processes existed for either project. Rather, both processes existed only in the heads of the involved disciplines. These teams produced a series of design and analysis documents such as CAD models, spreadsheets, and text files, but they did not formally describe the dependencies between them. The ability to formally represent the dependencies between representations in the computer may have enabled the design teams to more quickly and accurately communicate their design processes to the other project participants.
- *Integrating these design processes and information:* Both projects had difficulty quickly and accurately integrating their information and processes. On the concert hall, constructing and integrating the deck attachments representation cost the decking detailer over 140 hours and required over $160,000 worth of field welding that might have been avoided with better integration. On the office building, the design team also found it difficult to maintain the integration of all of the various analysis representations. For example, to explore a variation on an atrium option required several weeks, and errors and inconsistencies between representations occurred along the way.
- *Improving these design processes and information:* Both projects had difficulty optimizing their multidisciplinary processes and information. For example, they could not communicate and integrate this process quickly enough to iteratively modify the slab and beam designs and minimize the size and number of deck attachment, and many deck attachments required more costly, time-consuming, and less safe field welding. On the office building the design team was unable to

sufficiently explore many configurations of skylight and atria layout to determine the optimal layout for the energy, daylight, cost, and productivity criteria they determined were important.

2.4. REQUIREMENTS: AEC TEAMS COULD BENEFIT FROM SIMPLE METHODS TO FORMALIZE AND MANAGE NARRATIVES

In Haymaker et al. (2004), we proposed that AEC professionals could have addressed the difficulties listed above by better formalizing and controlling their design processes. For professionals to work in this way, such methods should be adequately:

- *Generic*: To apply across many different AEC disciplines.
- *Expressive*: To describe the many types of information and dependencies professionals need.
- *Formal*: To enable the methods to be explicitly communicated to other AEC professionals and to be implemented in a computer.

Simple: To enable broad understanding, acceptance, and use by engineers.

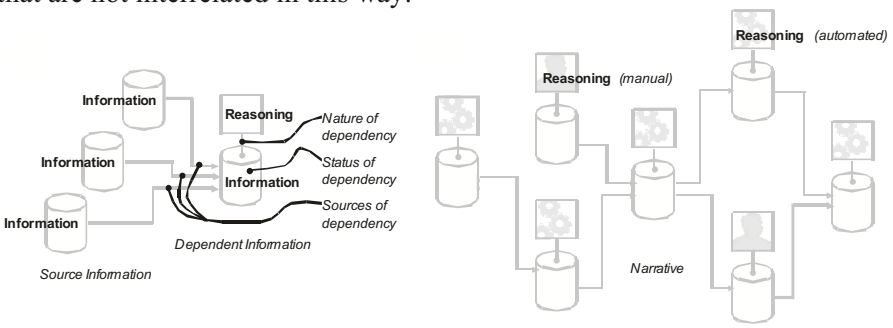
We proposed that MDA processes could be augmented by, if not founded on, simple, formal expressive, generic methods to construct information and specify its dependency on other information and by controlling the integration of this information as the project progresses. A formal Narrative could emerge as AEC professionals iteratively apply these methods. In Haymaker et al. (2004), we proposed the following methods, categorized as *representation*, *reasoning*, and *management* methods:

Representation: Engineers need adequately generic, expressive, formal, and simple methods to represent their task-specific information. There is a wealth of representations already developed by existing standards bodies (i.e., STEP, IFC, etc.) and private companies (i.e., AutoCad, Microsoft, etc.). However, these professionals still lack adequately expressive, generic, formal, and simple methods *to represent the dependencies between the information.* While acknowledging that the dependencies between information can often be cyclical – for example, the architect may revise the location of slabs or beams based on the number and size of deck attachments – this research investigates the conceptual simplicity of formalizing a project model as a directed acyclic graph (DAG) of information and their dependencies. AEC professionals will manage the cycles in the dependencies. From observations on the test cases, the majority of the information dependencies are one directional; in the spirit of simplicity we believed directed dependencies are worth exploring.

We formalized the following relationships and attributes to represent the dependency between information and its source information:

- *Sources*: The source information on which dependent information depends. For example, the Deck Attachments representation depends on the Steel Framing and Concrete Slabs representations.
- *Status*: The integration status of the information with respect to its source information. For example, when a Steel Framing or Concrete Slabs representation is modified, the Deck Attachments representation's status becomes Not_Integrated.
- *Nature*: The reasoning method (automated or manual) that constructs the dependent information from source information. For example, the decking detailer uses reasoning to construct the Deck Attachments representation from the Steel Framing and Concrete Slabs representations. Today, much of this reasoning is implicit, happening only in the heads of the AEC professionals.

Figure 1(a) diagrams this formalization of the dependency of dependent information on source information(s). Figure 1(b) shows how a formal Narrative can emerge from the iterative application of this method. In a Narrative, we often call the information a "Perspective", and the reasoning a "Perspector" to differentiate them from other representation and reasoning that are not interrelated in this way.



(a) (b)

Figure 1. Formalising the dependency between task-specific information; (a) formalizing the sources, nature, and status of the dependency of a dependent information on source information; (b) a Narrative emerges from the repeated application of the formalism described in A.

Reasoning: AEC professionals could use adequately expressive, generic, formal, and simple methods to define *the nature of the dependency*. In many cases computer algorithms already exist that help AEC professionals construct useful dependent information. For example, today it is possible to automatically construct plans or sections, and energy, daylight, or structural analyses. AEC professionals should be able to easily associate these "off-the-shelf" reasoning tools into Narratives.

In other cases, due to the unique nature of AEC projects, no such automated reasoning exists. For example, no algorithm exists for

constructing a Deck Attachments representation from Slabs and Beams representations, or for constructing an atrium design from site and requirements information. In these cases, AEC professionals should be able to easily define automated reasoning where possible.

In other cases, it is not possible, desirable, or economically feasible to define automated reasoning. Therefore AEC professionals should be able to specify that the nature of the reasoning is manual, perhaps also specifying that a computer tool (such as a CAD program) should be used to construct the dependent information. Figure 1(b) shows that a Narrative can contain a mixture of manual (denoted by a human icon) or automated (denoted by a gears icon) reasoning.

Management: Engineers could use adequately expressive, generic, formal, and simple methods to manage the integration of their information, so that they can iteratively construct their information and receive notification when the information on which they depend has been reconstructed. From the cases studies, it appears they should be able to easily and iteratively:

- Construct new representations, information in these representations, and dependencies between representations.
- Control the integration of their representations with respect to the representations on which they depend.

For example, the engineer responsible for the deck attachments representation should be able to easily construct the dependencies on the steel framing and concrete slabs representations, receive notification when these source representations are modified, and be able to (re)construct the deck attachments representation. Other engineers should be able to construct and control dependent representations of the deck attachments representation.

3. Points of Departure: Approaches to Duilding Information Modeling

This section discusses related efforts in research and practice in the area of representing, reasoning about, and managing building information, and in communicating design processes.

3.1. REPRESENTATION

Most AEC projects today rely on proprietary and task-specific information formats. This can result in serious interoperability difficulties when different project stakeholders adopt different proprietary and task-specific solutions. To address these difficulties, industry and government have initiated major efforts in the area of engineering data standards, including STEP (Standard for the Exchange of Product data (ISO 1994)) and IFC (Industry Foundation Classes (IAI 2004)). For example, the schema defined in IFC 2.X enables an

engineer to represent Ifcbeam features and an Ifcslab features. These standards enable computer programs to read and manipulate datatypes and are important for AEC productivity. However, the IFC contain over nine hundred standard datatypes, but the concept of a deck attachment is not explicitly considered in the current version. The ability to quickly extend or otherwise augment such datatypes is necessary (Fischer and Froese 1996). STEP, the IFC, and other standards such as XML – eXtensible Markup Language (W3C 2005) – provide a generic means for representing data, and can therefore be extended to cover new concepts. The AEC industry has a plethora of ways with which to represent AEC information. However, the cases show that engineers need to represent the sources, status, and nature of the dependencies between information. As currently formalized, these standard representation languages do not contain a simple, formal, explicit, and generic way to represent these dependencies.

3.2. REASONING

AEC professionals are using computer programs that automatically construct useful task-specific dependent information from source information in practice today. They are performing daylight analysis, energy analysis, structural analysis, and cost estimating, among other uses. Considerable research is devoted to improve on and extend these suites of pre-defined task-specific, automated design, and analysis programs. Generally, in such systems, a computer programmer with engineering knowledge programs task-specific reasoning that transforms source information into task-specific dependent information that is limited to the concepts formalized by the programmers. Other approaches to constructing task-specific representations of project information are more generic. Query languages and approaches (Date and Darwen 1993, Hakim and Garrett 1997) enable the automatic selection or limited transformation of information in a model into a view. Feature Recognition (Dixon and Poli 1995) identifies and formally represents instances of feature classes in a geometric model. Lou et al. (2003) investigates generic CAD query languages that enable engineers to query a model for geometric features. XSL (W3C 2005) is a language with which to transform XML representations into other XML representations. However, existing task-specific reasoning tools and generic query languages are not fully leveraged in AEC practice today. This may be in part because these projects have lacked a simple, generic, formal, expressive framework that enables engineers to quickly and accurately interrelate their task-specific representations and reasoning by their dependencies and control their integration as the project progresses.

3.3. PROJECT MANAGEMENT FRAMEWORKS

An increasing number of researchers and industry professionals are recognizing the need to formalize reasoning and management approaches that support information evolution for AEC projects. Some of these approaches (Eastman and Jeng 1999; Haymaker et al. 2000; Autodesk 2003; Sacks et al. 2004) develop reasoning and management that constructs and controls dependencies of information as views of a central model. This system architecture is finding its way into commercial applications (IES 2005). Others (Khedro and Genesereth 1994; Rosenman and Gero 1996; Mackellar and Peckham 1998; Sriram 2002) develop similar reasoning and management approaches that construct and control dependencies between information in a federation of predefined task-specific views. This system architecture is also finding its way into commercial applications. In both these central and federated model approaches, system programmers are generally required to construct the nature of the dependencies, and the narratives are thus predetermined.

Parametric techniques (Shah and Mäntyla 1995) enable professionals to define sets of related numeric or symbolic equations that can be solved to realize feasible geometric designs. Commercially available parametric modelers, such as CATIA, provide tools to assist engineers in generating 2D sketches from which 3D form features are parametrically generated and in specifying the assemblies of these form features parametrically with respect to the positions of other form features. Some systems employing parametric techniques are being commercially introduced specifically for the AEC industry, such as Tekla Xsteel, Autodesk Revit, Bentley's Generative Components, and Onuma's Object Genome System. These efforts parametrically define objects such as walls, windows, doors, and other geometric objects in terms of their properties and relations to other objects. While some successes are being reported within the context of single domains, parametric techniques are not being widely used in the AEC industry to integrate the work of multiple disciplines. This may be in part because, as currently formalized, these techniques have not adequately supported the multidisciplinary, constructive, iterative, and unique nature of AEC projects: They do not enable professionals to easily and formally construct new representations from information in other professionals' representations, and to control the integration of these representations as the project progresses. It may also be in part because they have lacked intuitive user-interfaces to communicate the dependencies between the information. Project scheduling systems, such as Primavera are task-focused, graph-based representations of a project. They represent precedence dependencies among tasks and are used to calculate issues such as project duration and critical paths. They do not contain an explicit representation of task-specific

information, nor do they represent or manage the nature and status of the dependencies between this information.

Current project modeling frameworks do not provide adequately simple, formal, generic, expressive methods that AEC professionals need to construct and control their MDA Narratives. Instead professionals are utilizing a hodgepodge of AEC systems without explicit connections, complicating the communication and integration of multidisciplinary information and processes.

3.4. OTHER RESEARCH IN FORMALIZING DESIGN PROCESSES

Issue-based information systems (Kunz and Rittel 1970) aim to enable designers to model and communicate their design rationale by recording the issues addressed, options considered, and the arguments both pro and con. These systems have used graph or tree based representations to structure and communicate these arguments. These concepts have been extended in various ways to, for example, enable information representations to be stored with each element in these graphs (Bracewell et al. 2004), however, to our knowledge, these systems have not yet emphasized the formal nature and status of the dependencies between these representations, and have not provided management processes to automatically manage these dependencies.

Design Structure Matrix (DSM) (Eppinger et al. 1990; Steward 1981; Austin et al. 2001) is a compact, matrix representation of a system or project. The matrix contains a list of constituent activities and the corresponding information exchange and dependency patterns. Most DSM's contain a list of tasks, and a formalization of the existence of an information dependency between the tasks. However, these representations do not contain an explicit representation of the actual information contained and processed at each task, nor do they represent the nature and the status of the dependencies between this information. Interestingly, the DSM was developed as an alternative to graph-based views of the project, in part because these researchers found the graph view non-communicative. We beleive today's better graph visualizations toolkits and larger display capabilities make graph-based visualizations of these dependencies worth reinvestigating.

Building Stories (Martin et al. 2005) capture design processes through an activity-artifact model, where activities embody the actions performed and the artifacts embody the information they create. Building stories are generally intended to capture the design process for use in explaining design rationale, and to be retrieved for use on subsequent projects. Building Stories do not contain a formal representation of the status and nature of the dependencies between information, and the authors have not emphasized their use as an active design integration tool.

4. Formalizing and Implementing MDA Narratives

This section presents conceptual and implemented Narratives and discusses the potential communication, integration and optimization benefits.

4.1. COMMUNICATION

Figure 2 presents a conceptual Narrative to formalize the cost-benefit analysis of the office building described in Section 2.1. The connections between representations are formally described. While the figure is perhaps daunting at first, we have found that once understood, this notation (Figure 1 is all there is to learn) simply and generically communicates important information about the dependencies between representations (the sources, status, and nature). As the figure shows, any reasoning node in this Narrative can itself be decomposed into a Narrative. Such decomposition aids in the thought process when constructing a Narrative, as well as the communicative ability of a composed Narrative.

Figure 4, later in this paper, shows an initial framework, described in Haymaker et al. (2004b) in which AEC professionals can quickly compose representation and reasoning into Narratives. Figure 5(a) shows the implementation of the Find Deck Attachments Narrative, and Figures 5(b) and 5(c) show the implementation of a Narrative (also described in Haymaker et al. 2004b) that automatically analyzes the concert hall ceiling for structural cantilever conditions. This implementation of our framework is somewhat limited, in that it runs on a single machine and handles only geometric representations and reasoning.

In Stanford's CEE 111: 3D Modeling plus analyses, students are asked to form teams and devise a project that uses at least two computer analyses tools to answer a question about a real world problem. This year, we asked students to describe the processes and information they are using for their project in terms of a Narrative. While the class is not yet to the mid term, we have found that the Narratives help to very clearly communicate the students planned process to other team members, to the remainder of the class, and to the professor. Figure 3 shows and describes one student Narrative.

4.2. INTEGRATION

Decomposing reasoning into sub Narratives can conceptually go to a very low level. In Haymaker et al. (2004b; 2004c), we show that the deck attachments representation (described in Section 2.1) can be constructed from the Concrete Slabs and Steel Framing representations, Figure 3, by analysing the beams and slabs to generate the deck attachments.

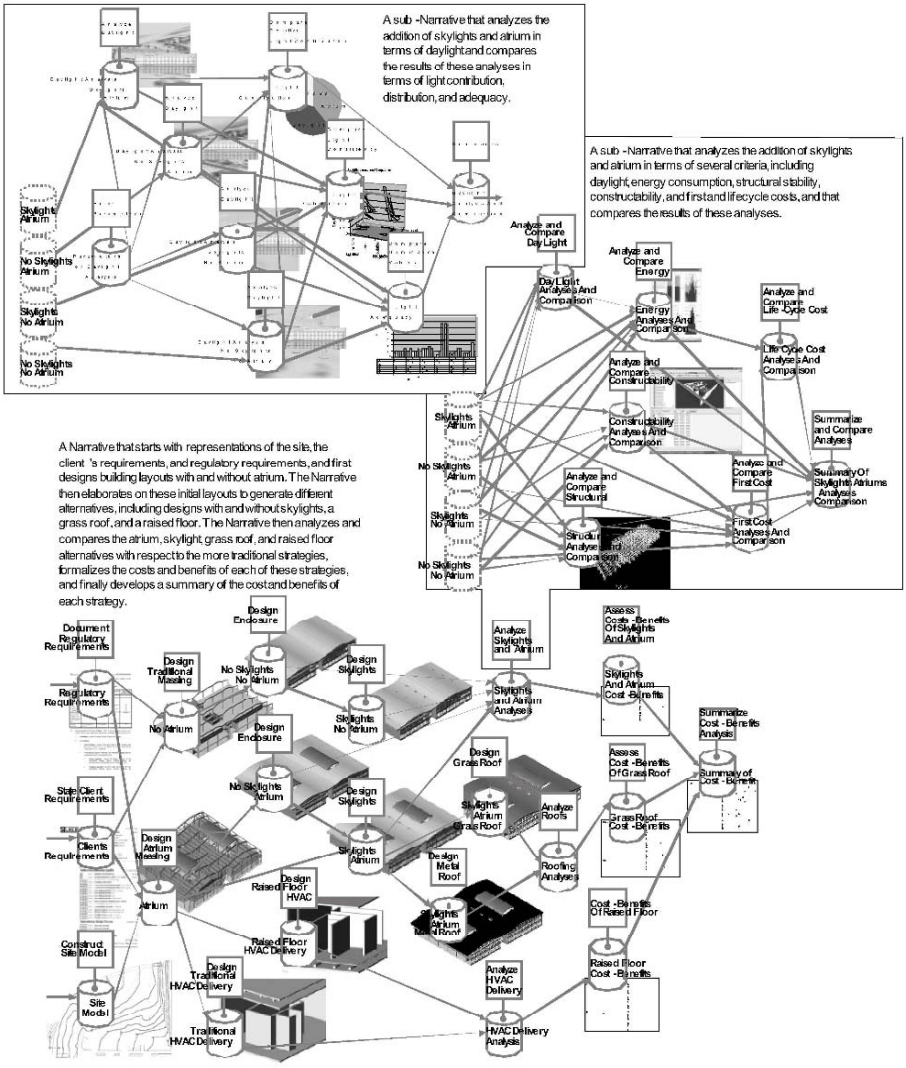


Figure 2. A conceptual Narrative to formalize a cost-benefit analysis.

When source information, such as concrete slabs, steel beams, ceiling panels, or ceiling panel supports representations are modified, the framework notifies dependent representations that they are no longer integrated. The engineer can then automatically integrate the dependent representation. Figure 6 provides some evidence for the ability of Narratives to improve the accuracy of integration. Figure 7 provides some evidence for the ability of Narratives to improve the speed of integration.

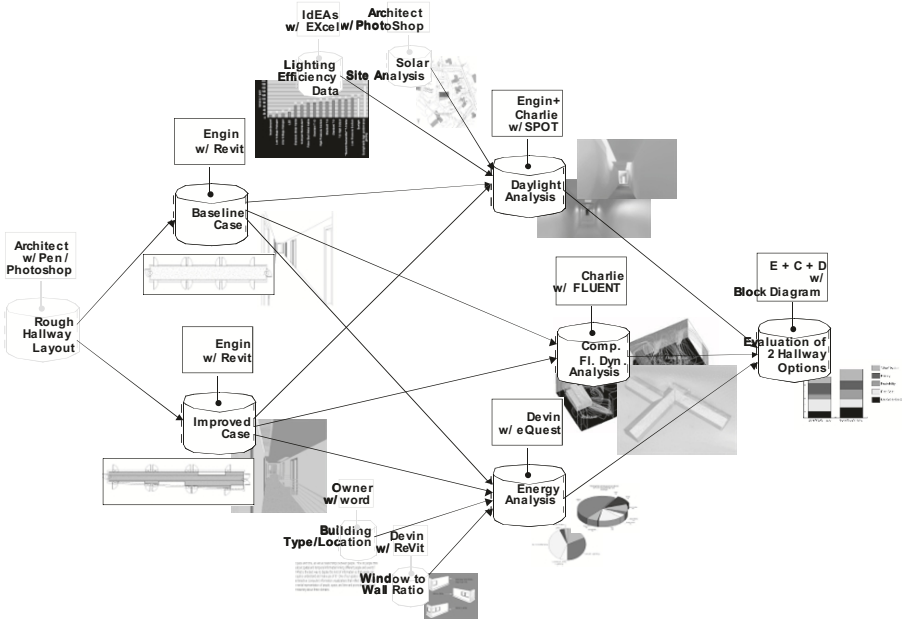


Figure 3. A Narrative designed by a group of students in Stanford University's CEE 111: 3D Modeling Plus Analyses, to describe his group's class project. Starting with an architect's initial rough sketch for a project, the students will explore different hallway configurations for their daylight, ventilation, and energy impacts. The students will then take the results of these analyses, and compare the designs for their integrated performance. At the time of writing, the students were just planning their design processes. Therefore, no analyses had yet been completed, so the images at each Perspective were taken from other projects.

4.3. IMPROVING DESIGNS

According to the architects in the case study described in Section 2.1, the project should "combine aesthetic rhyme with architectural reason, environmental sensitivity with operational efficiency, the diverse needs of individual employees with the scale and flexibility required by a growing company" (Bay Area Council 2000). The question remains: to what extent did they achieve these goals? Figure 8 shows a partial conceptual Narrative to measure these overall goals on six axes of a spider diagram. Each goal is measured by an interrelated set of sub Narratives.

For example, this architect describes environmental sensitivity in terms of several sub goals related to: access to fresh air, indoor air quality, integration with surroundings, energy, site, material flows, water, and access to light. Further sub Narratives would measure each of these sub goals.

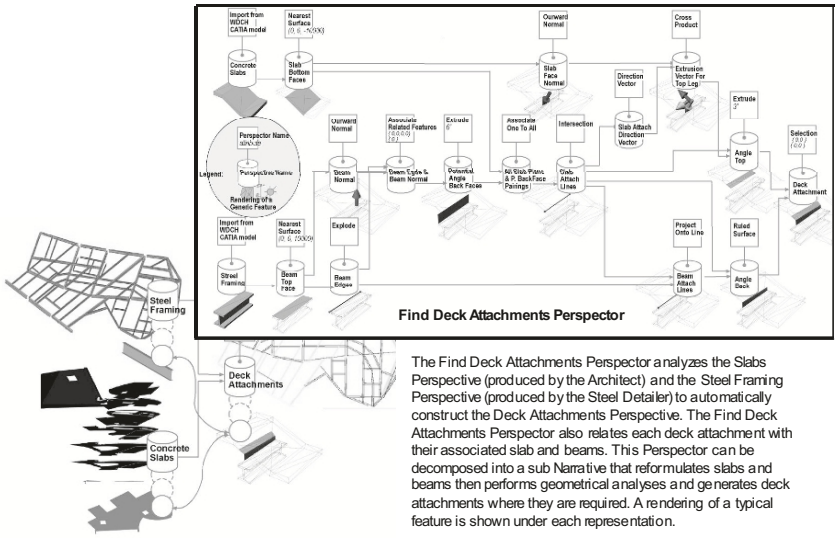
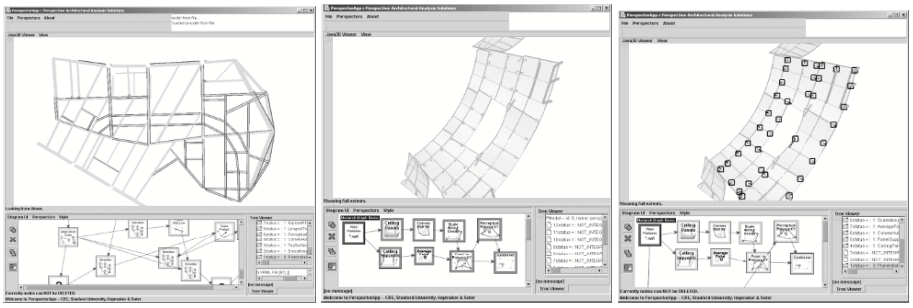


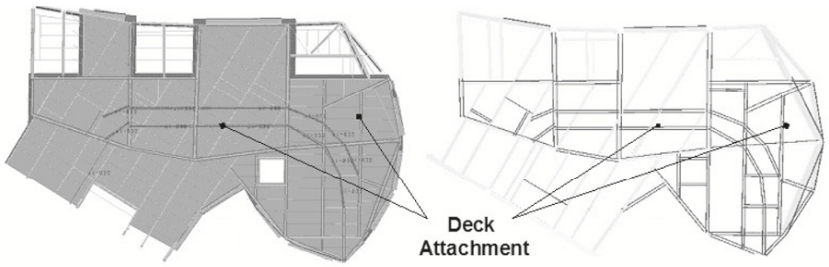
Figure 4. Applying Narratives to the Deck Attachment test case.



(a) (b) (c)

Figure 5. Our initial prototype for a Narrator that enables engineers to quickly connect reasoning and representations into Narratives. (a) The framework used on the deck attachment test case. (b) The framework was also used on a Narrative that analyzed the concert hall ceiling system for cantilevered conditions before integrating the Narrative, and (b) after integrating the Narrative.

For example, the architect describes energy efficiency in terms of: embedded energy in materials, building energy use, people and transit, renewable resources, and construction processes. Ultimately, these Narratives interweave other types of representation and reasoning, such as CAD drawings describing design options, analysis data describing energy calculations, and other types of representations. Ideally, the connections between these representations could all be formal. Modifications to any information could propagate through the Narrative, reflecting any changes to the overview of the six goals of the project.



	WDCH	Perspectors	
Accuracy	**86 Deck Attachments Required**		Significant improvement over current practice is possible. Further improvement possible with additional Perspectors.
Correctly Identified	**0**	**84**	
False Positives	**0**	**28**	
False Negatives	**86**	**2**	
Completeness			
Amount of detail			Automation could make creating more useful additional detail cost-effective.

Figure 6. Evidence for accuracy of integration: The left drawing shows the as-built Deck Attachments on a portion of the concert hall. AEC professionals failed to shop weld any of these deck attachments due to integration difficulties. The right drawing shows that we were able to automatically identify and design over 98% of the required deck attachments, making shop welding far more likely. The false positives were because the Narrative designed deck attachments on "stiffener beams" which have short spans, and do not require deck attachments. The Deck Attachment Narrative could be improved to eliminate the stiffener beams automatically, or the designer could remove the stiffener beams manually.

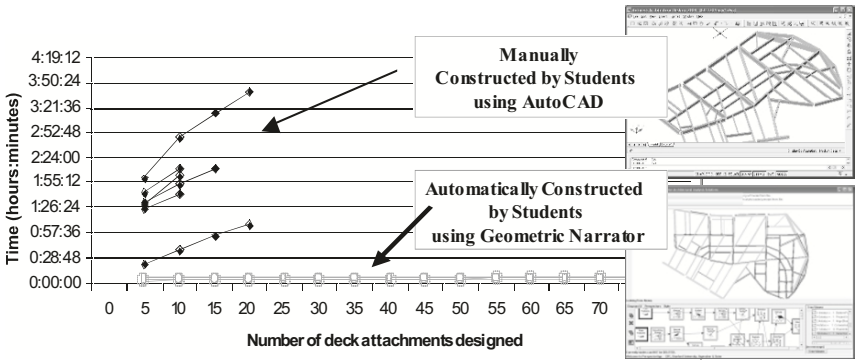


Figure 7. Evidence for speed of integration: Students of varying CAD skills were asked to manually design deck attachments for the steel and concrete designs using AutoCAD. Students were then asked to design deck attachments automatically, using the Narrator and the predefined Find Deck Attachments Narrative. The graph illustrates significant time advantages are possible using the Narrator.

This could enable far more options to be explored and analyses to be performed. When formal, automated connections are not possible, AEC professionals could continue their current practice of using manual

connections, but with help from the computer for communicating and integrating these Narratives and the status of their integration.

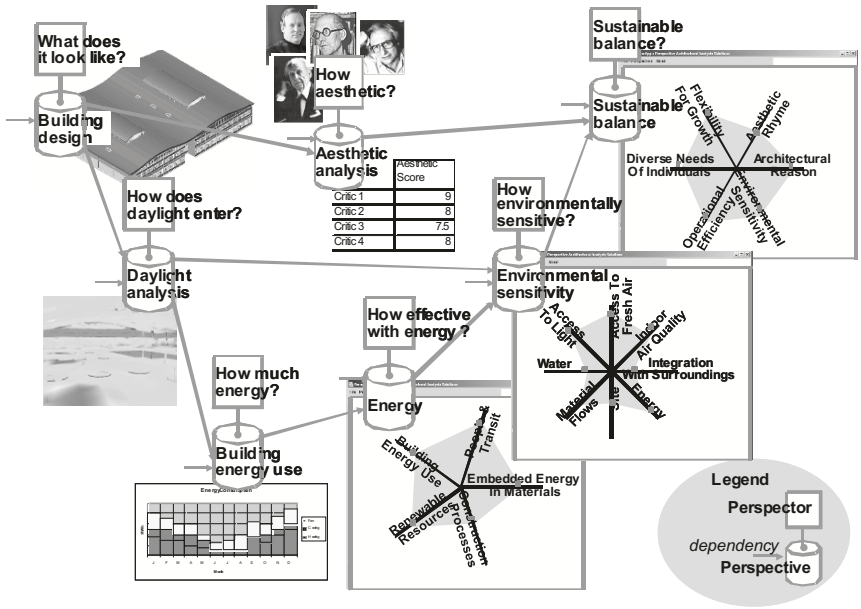


Figure 8. A partial conceptual Narrative to measure a project in terms of goals.

For example this Narrative shows that the measurement of aesthetics could simply involve asking four human design critics for their opinions. Our intuition, although not yet tested, is that Narratives can help designers collaboratively explore and analyze more options more quickly, and this will lead to improved designs.

5. Ongoing Work

We are currently extending the research in the following ways:
User Interface and immersive information environments: Narratives contain a great deal of information. Enabling AEC professionals to visualize and interact with the representations and the dependencies in a fluid manner is critical to enabling professionals to understand and make informed MDA decisions. We are working on more intuitive graph-based interfaces, and deploying the framework in the CIFE I-Room, to enable vastly improved user interaction with the Narratives. Figure 9 mocks up the proposed Narrator (a tool for constructing and controlling Narratives) in the CIFE I-Room. Users can iteratively view and construct the dependencies of the Narrative on the center screen, while they modify and view source and

dependent representations on the other screens. The figure shows a scenario where different professionals iteratively modify the building geometry and automatically receive feedback as to the design performance based on their multidisciplinary goals of the project.

Enable distributed representation and reasoning: AEC projects are distributed, often globally. Connecting the geographically distributed representations and reasoning of many disciplines is an important benefit that the federated architecture of Narratives can provide. We are investigating web-based technologies to allow Narratives that can be distributed over the web.

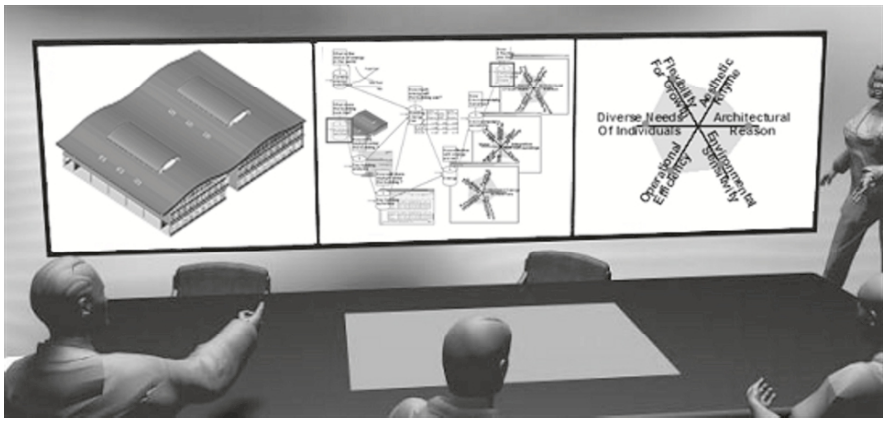


Figure 9. A mock-up of the Narrator in the I-Room. In this scenario, the team is iteratively modifying a design of the building (the left screen) as they attempt to achieve and exceed their project goals (right screen). The Narrative (center screen) describes and manages the dependencies of several task-specific design and analysis data models.

Incorporate any data types and reasoning: The test cases show that the types of representation and reasoning required on AEC projects are very diverse. Narratives must be able to integrate these diverse methods of representing and reasoning about AEC information in a simple, generic, expressive, yet formal framework. We are investigating ways to build a framework that is agnostic to types of representation and reasoning beyond the simple formalization of dependency described in this paper.

6. Conclusion

Today's methods for interoperable modeling and simulation are not working as effectively as they must to enable truly integrated design. AEC professionals need a toolset with which to effectively construct and integrate their information and processes. Given the multidisciplinary, constructive,

iterative, and unique nature of AEC projects, this proposed toolset will need to be flexible to evolve with practice and technology. This paper reports on ongoing work to formulate and validate a framework and language called Narratives that we hope can provide this power and flexibility, and discusses the benefits of Narratives for communicating, integrating and improving multidisciplinary design and analysis processes.

Ideally, many of the connections in Narratives could be formal and automated. Modifications to any representation could rapidly propagate through the Narrative, reflecting any changes as impacts on the goals of a project. When formal, automated connections are not possible, AEC professionals could continue their current practice of using manual connections. The transition to formalized and supported Narratives can be evolutionary and helpful, incorporating today's AEC computer tools; they are not meant to provide constraining meta-solutions that replace individual know-how and creativity. They are intended to integrate our greatest advances in information technology with the collaborative and creative human process of design and innovation.

Acknowledgements

Martin Fischer, John Kunz, and Ben Suter, William McDonough Partners, MA. Mortenson Company, Gehry Partners, Walt Disney Imagineering, Columbia Showcase, Martin Brothers/Marcowall Inc., Consolidated Contractors Company, and the many other firms and individuals from the Center for Integrated Facilities Engineering have contributed greatly to this work.

References

Austin S, Steele J, Macmillan S, Kirby P and Spence R: 2001, Mapping the conceptual design activity of interdisciplinary teams, *Design Studies* **22(**3): 211-32.

Bay Area Council: 2000, Environmental Building Design, (901 Cherry Offices) at Gap Inc., *Best Practices*, Bay Area Council, Available Online: http://bacqube.bayareacouncil.org/bp/bestpractices/bp185.html

Bracewell, RH, Ahmed, S and Wallace, KM: 2004, DRed and design folders: A way of caputuring, storing and passing on - knowledge generated during design projects, *Design Automation*, *ASME*, Salt Lake City, USA,

Date, CJ, and Darwen, H: 1993, *A Guide to the SQL Standard, Third Edition*, Addison-Wesley Publishing Company, Inc.

Dixon J and Poli, C: 1995, *Engineering Design and Design for Manufacturing*, Field Stone Publishers, MA.

Eastman, C and Jeng, T-S: 1999, A database supporting evolutionary product model development for design, *Automation in Construction* **8**(3): 305-33.

Eppinger, S, Whitney, D, Smith, R and Gebala, D: 1990, Organizing the tasks in complex design projects, *Design Theory and Methodology*, pp. 39-46.

Fischer, M and Froese, T: 1996, Examples and characteristics of shared project models, *Journal of Computing in Civil Engineering* **10**(3): 174-182.

Hakim, MM and Garrett Jr. JH: 1997, An object-centered approach for modeling engineering design products: Combining description logic and object-oriented models, *AI EDAM* **11**: 187-98.

Haymaker J, Fischer M, Kunz J and Suter B: 2004, Engineering test cases to motivate the formalization of an AEC project model as a directed acyclic graph of views and dependencies, *ITcon* **9**: 419-41.

Hollings J: 2004, *A Managed Environment for Plants*, Bentley Systems, Incorporated. Available Online: ftp://ftp.bentley.com/pub/outgoing/Bentley_Plant_Managed_Environment_White_Paperpdfhi.pdf

IAI: 2003, *Industry Foundation Classes*, *Version 2X2*, International Alliance for Operability. Available Online: http://www.iai-international.org/

IES: 2005, *Integrated Environment Solutions*. Available Online: http://www.iesve.com/

ISO: 1994, 10303-1: *Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 1*: Overview and fundamental principles, International Standards Organisation.

Khedro, T and Genesereth, MR: 1994, The federation architecture for interoperable agent based concurrent engineering systems, *International Journal on Concurrent Engineering,Research and Applications* **2**:125-131.

Kunz, W and Rittel H: 1970, Issues as elements of information systems. *Working Paper No. 131*, Institute of Urban and Regional Development, University of California at Berkeley, Berkeley, California, 1970.

Lou, K, Subramaniam, J, Iyer, N, Kalyanaraman, Y, Prabhakar, S, Ramani, K: 2003, A reconfigurable, intelligent 3D engineering shape search system Part II: Database indexing, retrieval, and clustering, *ASME DETC 2003 Computers and Information in Engineering (CIE) Conference*, Chicago, USA.

MacKellar, B and Peckam, J: 1998, Multiple perspectives of design objects, *in* JS Gero and F Sudweeks (eds), *Artificial Intelligence in Design*, Kluwer Academic Publishers, pp. 87-106.

Martin, M, Heylighen, A, Cavallin, H: 2005, The right story at the right time, *AI and Society* **19**(1): 34-47.

Rosenman, MA and Gero, JS: 1996, Modeling multiple views of design objects in a collaborative CAD environment, *CAD, Special Issue on AI in Design* **28**(3): 207-21.

Shah, J and Mäntyla, M: 1995, *Parametric and Feature-Based CAD/CAM*, Wiley and Sons Inc., New York, USA.

Sriram, DR: 2002, *Distributed and Integrated Collaborative Engineering Design*, Sarven Publishers.

Steward, Donald V: 1981, The design structure system: A method for managing the design of complex systems, *IEEE Transactions on Engineering Management* **28**: 71-74.

W3C (2005). *Extensible Markup Language*. Available Online: http://www.w3.org/XML/

ENHANCED DESIGN CHECKING INVOLVING CONSTRAINTS, COLLABORATION AND ASSUMPTIONS

Ontology-supported Rationale for Collaborative Argumentation

JANET BURGE, VALERIE CROSS, JAMES KIPER, PEDRITO MAYNARD-ZHANG
Miami University, USA

and

STEPHAN CORNFORD
NASA Jet Propulsion Laboratory, USA

Abstract. The design process involves selecting between design alternatives based on the requirements and constraints defining the system being developed. The design alternatives and the reasons for selecting one over another form the Design Rationale (DR) for the system. This information represents the designers' intent and can be used to evaluate the design alternatives and determine the impact on the design if requirements and constraints change. Here we introduce the Ontology-supported Rationale for Collaborative Argumentation (ORCA) system which uses DR to capture and evaluate alternatives for Engineering Design. Moreover, we motivate and demonstrate our system using a NASA spacecraft design study.

1. Introduction

The design process has been described as "a process of establishing which of several alternative ways (and with what tools) things could be done, which of these is most promising, and how to implement that choice, with continual reviews, additions and corrections to the work — designing" (Hubka and Eder 1996). This definition is not complete — it does not indicate how the alternatives are generated. Also, while indicating that the "most promising" alternatives should be used it does not indicate what that means operationally. Tong and Sriram's definition (1992) emphasizes the requirements placed on the design — the need to conform to a specification, meet certain criteria (such as performance requirements and resource constraints), and work within constraints such as technology and time

J.S. Gero (ed.), Design Computing and Cognition '06, 655–674.
© 2006 *Springer. Printed in the Netherlands.*

limitations. The conformance with requirements and constraints is what makes an alternative "most promising."

The ability to evaluate alternatives, particularly in relation to the requirements and constraints driving the design, is crucial to the design process. This evaluation needs to account for many factors including the *uncertainty* and *risk* involved in the decision-making process. The ability to perform a rigorous analysis of all the factors affecting the choice of design alternatives is vital to developing a good design.

Much of the information needed to perform a comprehensive assessment of design alternatives can be found in the Design Rationale (DR) for the alternatives. The DR describes the design decisions, the alternatives considered, and the argumentation describing the advantages and disadvantages of each candidate alternative. This information offers a rich view of both the product and the decision-making process by providing the designer's intent behind the decision (Sim and Duffy 1994). Recent work on Software Engineering Using RATionale (SEURAT) (Burge 2005; Burge and Brown 2006) demonstrated using rationale to evaluate design alternatives and to perform "what-if" inferencing to show the impact on the design if requirements, assumptions, or system goals change during or after the design process. There has been resistance in the past to rationale systems because the capture process is thought to be too time consuming and expensive but Bracewell's recent work with the DRed tool showed that it is possible to design a DR-capture tool that engineers feel helps, rather than hinders, the design process (Bracewell et al. 2004).

Applying rationale to the analysis of engineering design alternatives poses a number of interesting research questions. How can the rationale be used to evaluate design alternatives? How can we use the rationale to combine inputs from multiple designers? How can we use the rationale to assure consistent application of design requirements, constraints, and criteria throughout the design?

Earlier work on the SEURAT system demonstrated how rationale can be used to evaluate design alternatives based on the system requirements, design criteria, design dependencies, and assumptions captured in the rationale. That work used an argument ontology (Burge 2005) to support propagation of changing priorities for the criteria used in making these design decisions and also performed inferencing to detect incompleteness and inconsistency in the rationale. We are extending this work to the field of engineering design and enhancing it with more rigorous algorithms for the evaluation; extending the rationale representation to support collaboration, uncertainty, risk, and capture of design and resource constraints; and using design product and design process ontologies to support classification and visualization of design alternatives. In this paper, we describe the first steps taken in our development of this enhanced approach, known as

Ontology-supported Rationale for Collaborative Argumentation (ORCA).We also illustrate how our approach will be applied to the spacecraft design domain.

Section 2 motivates our research by presenting why it is important to capture and use the design rationale. Related work in design rationale and ontologies in engineering design is examined in Section 3. Section 4 overviews the ORCA system. Section 5 details the rationale representation for ORCA, and Section 6 describes how ORCA performs inferencing using this rationale to support the designer. We present conclusions and future research efforts in Section 7.

2. Motivation for Capturing and Using DR

Design is a complicated activity requiring analysis of the problem being solved and creation (synthesis) of a solution. It has been referred to as "an ill-structured activity requiring intelligence" (Brown 1993). Design, or designing, can refer to many types of design ranging from the routine to the creative (Brown and Chandrasekaran 1985).

Designing large, complicated systems is a collaborative process requiring input from many designers with different areas of expertise. These designers are responsible for generating and evaluating the design alternatives that will eventually compose the final integrated design. This evaluation process is made more challenging by the need to meet multiple goals and constraints. Alternatives are evaluated and selected based on the requirements for the system being designed and their ability to meet various design criteria such as performance, cost, and reliability. The evaluation must also incorporate assumptions made about the system being designed, the environment the system will operate in, the capabilities of technology, the experience of the people interacting with the system, and more. Some of these assessments have quantitative measures while others rely on the experience and subjective judgment of the designer. Still others may just be 'guesses' which act as a placeholder for values yet to be provided. In collaborative design, these subjective assessments often result in disagreements between the designers that need to be resolved in order to make a decision. The process is further complicated by the fact that some design criteria conflict with each other. These conflicts require tradeoff analysis when making the decisions. Decisions may also involve alternatives that are dependent on, or in opposition to, other alternatives under consideration in the design.

To motivate this work, we are using a design study produced by NASA's Jet Propulsion Laboratory (JPL) that examines the design of the FAIR-DART telescope and spacecraft (Oberto 2002). This telescope is intended for deployment in the year 2014 and is intended to study the behavior of interstellar gas and dust. The designers in the study were members of JPL's Advanced Projects Design Team ("Team X"). The goal of the study was to

identify critical areas, risks, and initial requirements and constraints for the spacecraft. We are using this study, along with additional information from NASA designers, to obtain examples of rationale to test our approach.

To support the design process, we believe the relationships between the design decisions, design alternatives, design requirements, and other design criteria involved in the decision-making can best be captured as part of the design rationale for the system. This rationale can then be used to evaluate the design alternatives and to re-evaluate the design alternatives when any of the contributing factors change during, or after, the design process. The rationale also serves as a description of the designers' knowledge about the domain and how it influences their selection of alternatives.

3. Related Work

3.1. DESIGN RATIONALE

DR research has concentrated on the representation, capture, and use of the rationale. Lee (1997) has written an excellent survey describing DR work in the engineering design domain and how it can be represented and captured. Here we will concentrate on work that investigates the *uses* of rationale.

There are a number of different ways that design rationale is used. Some systems only support retrieval of the rationale; how it is used after being retrieved is up to the designer. Some retrieval systems offer the ability to ask questions about the design and/or rationale (Garcia et al. 1993; Gruber 1990). The Engineering History Base System (Taura and Kubota 1999) uses constraints to provide teleological and causal explanations of the designers thought processes.

Some systems support retrieval and also offer the ability to check the rationale and/or the design for consistency and/or completeness. KBDS (King and Bañares-Alcantara 1997) uses keywords to check the consistency of IBIS networks that contain the rationale. C-Re-CS (Klein 1997) performs consistency checking on requirements and recommends a resolution strategy for detected exceptions.

Rationale has been used to support collaboration and integration by a number of systems. An early use of the IBIS notation was to use argumentation to structure discussion during design meetings (Conklin and Burgess-Yakemovic 1995). IBIS is used by QuestMap, a groupware system, to capture organizational memory (Conklin 1996). This was also the basis of the Compendium methodology (Conklin et al. 2001). The SHARED-DRIMS system (SHARED-Design Recommendation-Intent Management System) uses a DR ontology (DRIM) to capture design rationale for conflict mitigation (Peña-Mora et al. 1995). WinWin (Boehm and Bose 1994) supports collaborative work in the software engineering domain. HERMES

(Karacapilidis and Papadias 2001) captures discourse as argumentation and checks users' preferences for inconsistencies automatically throughout the elicitation process.

Experiments have demonstrated the usefulness of rationale for requirements analysis and design (Conklin and Burgess-Yakemovich 1995), design evaluation (Karsenty 1996), and software maintenance (Bratthall et al. 2000; Burge and Brown 2006).

3.2. ONTOLOGIES FOR ENGINEERING DESIGN KNOWLEDGE

Engineering design is a knowledge-intensive activity. Designers' minds often serve as repositories for vast amounts of knowledge which enable them to make intelligent design decisions. For any design activity, the task of knowledge management becomes critical. An increasing number of researchers (Eris et al. 1999; Fowler et al. 2004; Japikse et al. 2003; Kitamura et al. 2004; Lin et al. 1996) are pursuing the development of engineering design ontologies, motivated by the benefits of knowledge sharing, reuse, and a standard engineering design language.

Researchers at Cambridge University have developed the Engineering Design Integrated Taxonomy (EDIT) (Japikse et al. 2003). EDIT consists of several taxonomies with concepts and relationships developed by interviewing engineering designers. Some of its specified purposes are as an ontology for engineering and as a tool for cataloging and retrieving design knowledge. Other researchers (Kitamura et al. 2004) have developed an ontological framework to systematically express functional knowledge that is clearly applicable across domains. Previous researchers (Chandrasekaran 1993) have noted the importance of functional models as one piece, but not all, of the designer's intentions, or design rationale.

4. Overview of Our Approach

The SEURAT system (Burge and Brown 2004; Burge and Brown 2006) demonstrated how rationale, and the ability to inference over the rationale, could be used to support software maintenance. While SEURAT focused on rationale for software, much of the rationale representation and inferences used in SEURAT are extensible to other domains as well.

Here we describe a new system for the capture and use of rationale for engineering design: the *Ontology-supported Rationale for Collaborative Argumentation (ORCA)* system. ORCA uses a series of ontologies to describe the design process, product, and criteria. The process and product ontologies are new to the ORCA system while the design criteria ontology is an extension to the argument ontology developed for SEURAT (Burge 2005). Figure 1 shows the ORCA architecture.

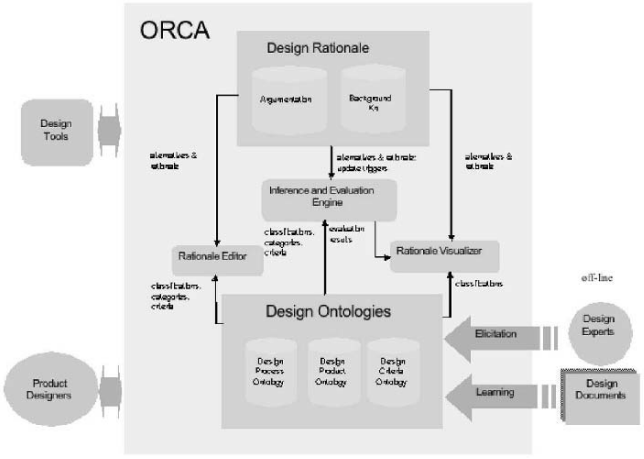


Figure 1. ORCA System Architecture.

To drive and motivate ORCA development, we use a description of the rationale (including assumptions and risks) involved in the design of a spacecraft propulsion system for the FAIR-DART mission. We used this information to extend the RATSpeak representation used in SEURAT to support engineering design rationale. SEURAT serves as our initial platform for developing and evaluating the new representation and inferences required. We have also extended SEURAT to handle design collaboration which is applicable in SEURAT's domain of software engineering as well as for ORCA's domain of engineering design. The remainder of this document will describe these extensions as well as present our plans for future ORCA development.

5. Rationale Representation

A key decision in any design rationale approach is how the rationale will be represented. The representation needs to provide the ability to specify the key elements in the design reasoning that are used to make design decisions. Our goal is to provide a representation that is expressive enough to convey the intent of the designer yet structured enough to be used in inferencing.

The rationale in ORCA consists of three main components:

- *Design rationale argumentation* – based on the RATSpeak representation developed for SEURAT with additional extensions needed to support constraints and collaboration.
- *Background knowledge* – includes the two types of background knowledge used in SEURAT, the tradeoffs and co-occurrence relationships, and also a set of designer profiles described by a *designer profile ontology*. In addition, the background knowledge includes a set of *design contingencies* that apply to different maturity levels of design

and need to be taken into account when evaluating how well the design meets its constraints.

- *Design Ontologies* – includes three types of design ontologies used in ORCA. The *design criteria ontology* is an extension of the argument ontology used in SEURAT. A *design product ontology* describes the major components of the system being designed. A *design process ontology* describes the design process being used.

The following sections describe these three components and what has been implemented.

5.1. DESIGN RATIONALE ARGUMENTATION

We have chosen an argumentation format for our rationale because argumentation is often the best means for expressing the advantages and disadvantages of various design options being considered. Each argumentation language has its own set of terms, but the basic goal is the same: to represent the decisions made, the possible alternatives for each decision, and the arguments for and against each alternative.

The RATSpeak representation focuses on design decision problems (known as *decisions* in the rationale) and the *alternatives* for solving them. Each alternative can have *arguments* for and against it. These arguments can refer to system *requirements* (violates, satisfies, addresses), *assumptions* about the environment the system will be operating in (supports, denies), dependencies with other alternatives (presupposes, pre-supposed-by, opposes, opposed-by), claims that the alternative meets or does not meet some *design criteria* (supports, denies), and arguments disputing or supporting other arguments (supports, denies). The design criteria, which capture non-functional requirements (NFRs) for the system, are stored in an argument ontology that serves as the basis of the ORCA Design Criteria Ontology. The rationale also includes any *questions* that need to be answered before an alternative can be fully evaluated and selected.

In the current version of ORCA, the argumentation has been extended by adding *constraints* to the rationale to show where they should be taken into consideration. In addition, we have enhanced the assumption representation to capture temporal information, and have implemented the ability to specify arguments for and against other arguments (this was a feature that was supported in RATSpeak but not implemented in SEURAT). Figure 2 shows the elements in the extended RATSpeak language.

5.1.1. Design Constraints

The design process involves creating a description of an artifact that satisfies its constraints (Brown 1993). In the domain of spacecraft design, critical constraints include power, mass, and cost. These constraints factor

prominently in many of the decisions described in the FAIR-DART study (Oberto 2002).

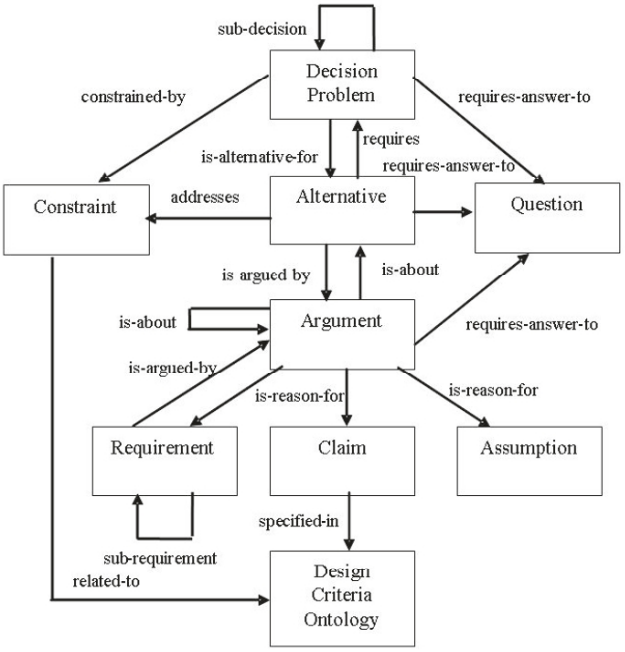


Figure 2. RATSpeak Argumentation Relationships.

Constraints are explicitly represented in the ORCA rationale. They are arranged in a hierarchy that parallels the design sub-systems given in the design product ontology by relating each constraint to the portion of the system to which they apply. For example, not only is there an overall mass constraint for the mass of the entire spacecraft, there are also individual mass constraints for each of the subsystems comprising it. Constraints are also associated with the elements in the design criteria ontology that affect the system's ability to meet the constraint. For example, reducing mass is an element of the design criteria ontology that is mapped to the related constraint.

Constraints played a big role in the spacecraft design problem. One example that we have stored as rationale in ORCA is the mass constraint associated with the choice of propulsion system. In the rationale, this is incorporated in several places. First, there is the constraint itself. The total mass of the propulsion system can not exceed 46,000 kgs. That constraint is associated with the Propulsion component in the Design Product ontology. That association is stored inside the constraint itself. The constraint can also be associated with any decisions that it affects and the alternatives that need to be decided between. For the propulsion mass constraint, it is associated

with two decisions: Propulsion to Position L1 and Propulsion to Position L2. This indicates that any alternative chosen for those decisions must also include a relationship that indicates its relationship with the constraint (i.e., its mass). The selected alternative for the Propulsion to Position L1 decision was Hydrazine Thrusters. This alternative has a relationship to the mass constraint that indicates that the mass of that propulsion system is 80 kgs.

In addition to associating alternatives with constraints, ORCA also requires the specification of the type (*maturity level*) of design for each alternative. This associates the alternative with the applicable contingencies (for example, a new design is riskier, and requires greater contingencies when specifying mass, power, etc. than an existing design where the requirements have been verified). Contingencies are described in more detail in the section on background knowledge.

5.1.2. Collaborative Argumentation

Not everyone involved in the design process will always agree with the recommendations of their teammates. Each team member brings their own experience to bear on the alternatives that they propose and their reasons for selecting one over another. For example, once the telescope has been deployed, a cold gas propulsion system is one way to avoid contamination of the optics (cold gas systems use the reaction force of the controlled release of inert gases, such as N2, to perform small attitude corrections. Thus, they have no plume that could cause contamination). One argument against the approach (Oberto 2002) was as follows:

"Previous cold gas systems have developed uncontrollable leaks."

A second expert, when asked about this, responded that this sounded like someone's opinion and that there have probably been many systems without uncontrollable leaks. The ORCA system will allow dialog such as this to be captured as part of the rationale so that it can be analyzed to see where areas of contention might be and how those should affect the evaluation of the different alternatives.

RATSpeak's already existing capability to represent arguments supporting or denying other arguments is implemented as part of ORCA. Each argument in RATSpeak is represented by at least two components:

- The argument itself. Each argument has an importance (how important is this argument in making the decision), an amount (how much the argument applies), and a plausibility (the certainty that this is a valid argument) specified. The values for these attributes are provided by the designer based on their experience. The contents of the argument stored in the rationale apply to one and only one decision.
- What the argument is about. This could be a claim (which then would map to an element in the design criteria ontology), an assumption, a requirement, or another alternative. These items can be used in multiple

places in the rationale (for example, the same requirement may be involved in multiple decisions). Arguments can also be about other arguments.

Arguments about other arguments indicate agreement or disagreement. For agreement, a simple supporting argument allows the supporting designer to indicate why they agree with the argument. For disagreement, the disagreeing argument is a duplicate of the original argument where the designer disagreeing can modify any of the parameters from the original argument (importance, amount, plausibility) and describe why they disagree.

The argumentation is kept to two levels: arguments about the alternative and any arguments for or against those arguments. This restriction simplifies the use of the arguments and avoids situations containing a long chain of arguments in which one disagrees with the original argument, another disagrees with the disagreement, and so on. This type of discourse can still be captured by multiple arguments agreeing or disagreeing with the initial argument causing the controversy.

To support collaboration among designers, the rationale is linked to the designers who created it for three of the rationale components: the decisions, the alternatives, and the arguments. As each element is created, the designer's name is attached. This will be done automatically in later versions of ORCA but is currently done manually from a list of designers. Each designer has a profile stored as background knowledge that describes their position in the organization, responsibilities, and area(s) of expertise.

5.1.3. Assumptions

Many design decisions, especially in cutting-edge design such as that practiced at NASA, involve making assumptions about the environment in which the system will operate and the technology available at the time of system deployment. The following example is taken from the design dialog occurring when NASA Jet Propulsion Laboratory (JPL) engineers were proposing the use of a cold gas propulsion system. This example shows some examples of assumptions made that reflect the risk involved in that design alternative.

> "A cold gas system can be built which will last the required time for this mission (about 6 years total, including 5 years at L2). Historically, cold gas systems have had lifetimes limited by leakage."

> "Nitrogen can be used as the cold gas propellant. It is understood based on customer input that a micro-layer of nitrogen gas frozen on the surface or in a cloud for a short period of time is not a problem."

The first excerpt assumes that technology (either newly developed or developed soon) will support building a system that was not viable in the past. The second excerpt states that a potential side effect of the type of

propellant proposed will not be a problem for the spacecraft. These assumptions contribute to the assessment risks and uncertainties associated with this choice and need to be explicitly documented and monitored to ensure that they continue to hold both as the design progresses and as the designed system is deployed and used.

The ability to capture assumptions and use them in evaluating design alternatives is a key feature of the ORCA system. The prevalence of assumptions as reasons for and against design alternatives for the FAIR-DART system serves to underline that importance. One type of assumption that is especially relevant in the spacecraft domain a temporal assumption that some event will happen or that some fact will be shown to be true by some point in time. The FAIR-DART design session was held in 2002 to discuss options for a spacecraft that would be completed in 2014. Clearly the technology available is expected to have evolved by 2014. The design process needs to design for the future and involves considering the use of technology that is not yet fully developed. In some cases these assumptions are stated in general terms and merely point out that this assumes a future development. In others the assumption explicitly states a cut-off date by which the technology development must be complete in order to be included in the final spacecraft design. RATSpeak has been extended to explicitly note where an assumption comes with a time limit so that the affected alternatives can be re-examined when the target date becomes closer.

5.2. BACKGROUND KNOWLEDGE

ORCA supports capturing *tradeoffs* and *co-occurrence* relationships between design criteria as background knowledge so that knowledge can be used to check for incompleteness and inconsistency in the rationale. In ORCA, an example of a co-occurrence is the relationship between the ability to operate at low temperatures and to have low power consumption (a device that needs to operate in low temperatures also will need to have low power consumption so that it does not generate excess heat). An example of a tradeoff is the tradeoff between minimizing contamination and maximizing reliability that occurs in the selection of different propulsion systems (propulsion systems that minimize contamination are often prone to leakage). Another is the tradeoff between increasing the precision of the orbit and decreasing the amount of fuel to be carried. The ORCA background knowledge also includes a description of design types, which indicate design maturity levels and contingencies for the organization developing the system, and a set of designer profiles to describe the design contributors.

Design contingencies are important in determining if the alternatives selected are going to meet the system constraints. Depending on the artifact

being designed and the team's previous experience, there may be uncertainty involved in the accuracy of estimates of factors such as mass and power consumption. This uncertainty is managed by allowing for contingencies. JPL has a set of design principles given in the FAIR-DART document (Oberto 2002) that describe how much should be added to each estimate prior to calculating the final result. For example, an inherited design would have a contingency of 15% while a new one would have a contingency of 30%. These contingencies are intended to ensure that a large collection of implementation uncertainties will be solved without requiring a re-design of the spacecraft or mission.

The designer profile provides supporting information that will help determine how much weight should be given to the arguments of the various designers involved in the decision-making process. This includes the designer's role on the project, level in the organization, level of expertise in the different aspects of the system (such as expertise in propulsion vs. power vs. thermal), and design experience both in the current organization and over the course of their career.

5.3. DESIGN ONTOLOGIES

An objective of this research is the development of an overall knowledge management framework for the capture, storage and retrieval of engineering design knowledge. This framework is supported by the use of ontologies. The basic components of an ontology are its *concepts* which describe sets of entities in a domain and its *relationships* which exist among the concepts. Concepts are described by their attributes and the relationships in which they participate. Two of the primary relationships used within ontologies are *taxonomic* and *associative* relationships. Taxonomies are organized as a sub-super concept tree structure, using "*isa*" relationships. Associative relationships connect concepts across the hierarchical structure. This structure provides a basis for navigating, browsing and searching the knowledge, and aids designers by making them aware of the kinds of available information and assisting them in formulating appropriate queries. Three engineering design ontologies are used in the ORCA system: the Design Product Ontology, the Design Process Ontology, and the Design Criteria Ontology.

For design rationale to be effective in assisting the design process, it must be set in the context of and be able to interact with engineering design knowledge. A designer undertakes a design process made up of numerous steps in order to design a product. The Design Product Ontology contains knowledge about the product being designed and the Design Process Ontology contains knowledge about the process used to design the product. The responsibility of an engineer is to design and specify a physical product

or system (component, sub-assemblies and assemblies) such as the spacecraft's instrument subsystem, or the hardware component of the instrument subsystem. Each alternative expressed in the DR relates to the product and/or its components and refers to the appropriate Design Product Ontology entries.

One may view a product concept from three different perspectives: structure or form, function, and behavior (Gero 1990). Design product form concepts correspond to the physical and abstract concepts that describe the product itself, such as the propulsion system of a spacecraft. The form view also represents the product's components and their relationships. Product function concepts describe the purposes of the products or components. The way a product form achieves its function is represented by product behavior concepts. For example, in our spacecraft example, the system consists of several components which include propulsion, thermal, telecom, ground systems, ACS (Advanced Camera for Surveys), CDS (Command and Data Systems), and structures. The propulsion system consists of two components: the propulsion system used to get the spacecraft into its initial position and the system used to maintain position after the telescope has been deployed. Each propulsion system contains attributes that describe various key parameters. Some examples include initial mass, attitude control, propellant type and propellant mass. Many of these attributes are associated with the needed product behaviors for achieving its functions as described by the requirements captured in the rationale.

The engineering design process information consists of the various tasks such as material selection or cost analysis undertaken at each stage of the design process, such as conceptual or detailed design. The Design Criteria ontology contains reasons for selecting one design alternative over another and is used to provide arguments for and against alternatives and to express common tradeoffs and co-occurrence relationships as described previously. These criteria are similar in nature to the dispositions described by Andreasen and Olesen (1990). The higher level criteria describe beneficial effects that are desirable when making design choices while the more detailed criteria can serve as disposition rules by indicating how a design alternative achieves the desired effect.

The design ontologies provide a formal structure to the various categories of knowledge and even the relationships among them. This structure provides a basis for navigating, browsing and searching the knowledge, and aids designers by making them aware of the kinds of available information and assisting them in formulating appropriate queries. The ontologies also support DR use by grouping alternatives to support evaluation calculations performed on a component or subsystem basis and the associating design stages and tasks to relevant DR entries.

The Design Criteria Ontology is currently the SEURAT Argument Ontology with some additional information. Additions and modifications to this ontology will occur over time as the system development progresses and more design documents are studied. The Design Product Ontology implemented is currently only a breakdown of the major sub-systems of the spacecraft being designed. This organization captures much of the rationale expressed in the preliminary design document but more product concepts, their attributes and relationships, as previously described in the example propulsion system, are to be added as project implementation continues. There currently is not a Design Process Ontology implemented although the system does allow the user to specify the development phase for each decision (requirements, analysis, design, implementation, and test).

6. Inferencing Over the Rationale

ORCA performs a number of different types of inference over the rationale. These fall into two categories: syntactic inference that looks for inconsistency and incompleteness in the structure and semantic inference that looks at the contents of the rationale to find contradictory arguments and to evaluate the level of support for each alternative (see Section 6.4 for alternative evaluation). Syntactic checks include reporting incomplete rationale such as having no arguments in favor of a selected alternative. Semantic checks include reporting tradeoff violations. A detailed description of the inferencing supported by our original system, SEURAT, can be found in Burge (2005). In the following sections we describe how we will use additional information captured in the rationale in our initial version of the ORCA system to provide enhanced support for decision-making for engineering design. We also describe our plans for using ORCA to assist in a more rigorous evaluation of the design alternatives.

6.1. ASSUMPTIONS

SEURAT uses assumptions as part of alternative evaluation along with the other arguments for and against each alternative. The designer can specify the assumptions used in making their decisions and also disable assumptions if they did not remain true or if the designer wanted to see the impact on the design if the assumption was to change. In ORCA, the assumption representation is extended to indicate which assumptions had a temporal component signaling that their applicability may change over time. In addition to the ability to disable individual assumptions, ORCA will provide the ability to filter out certain classes of assumptions, the first of which are those that may become invalid over time.

6.2. CONSTRAINTS

Constraints factor into the semantic inferences that check if the rationale is complete. The relationships between the constraints and the elements in the design criteria ontology will be used to check if the alternatives for a constrained decision have arguments for or against them that refer to those criteria. Constraints will also be used to query the rationale to obtain the impact of the selected alternatives on meeting these constraints. The designers can then use that information to evaluate the design's ability to meet the constraints. ORCA could perform these calculations automatically (such as by summing up the mass of all components in a sub-system) but it is probably not realistic that all components that affect a constraint such as mass or power consumption will be mentioned as alternatives in the rationale. The constraints are used to document the reasons for making the design decision and to ensure that selected alternatives address those constraints. An error is reported if a selected alternative for a constrained decision does *not* provide its impact on satisfying the constraint.

6.3. COLLABORATION

Combining contradictory arguments for and against alternatives that come from different designers is an interesting area of research. Currently, alternative evaluation in ORCA only considers the "first level" arguments – those for and against alternatives – rather than arguments about other arguments. Another possible way to handle contradiction would be to go with the argument from the designer with the most expertise in that design area, e.g. override the initial argument if someone who "outranks" the author of the original opinion disagrees with the original argument. Contradiction could also be handled by using all the arguments but assigning each a "credibility" based on the designer's expertise and using that as a weight in combining them into a single argument for or against the alternative. The method of combination may vary depending on the type of argument. For example, if one designer feels an alternative violates a requirement and another disagrees, a safer choice might be to adopt the lower rating since the negative impact on a design of selecting the alternative if it turns out to violate the requirement would be significant. We will leverage work in the emerging field concerned with using information about information sources to resolve conflicts, e.g. (Matzkevich and Abramson 1992; Cantwell 1998; Pennock and Wellman 1999; Maynard-Reid II and Chajewska 2001; Maynard-Reid II and Shoham 2001; Andéka et al. 2002; Maynard-Zhang and Lehmann 2003).

In any case, ORCA will analyze the rationale and indicate which selected alternatives were controversial so those can then be singled out for further investigation. We do not plan to replace the valuable and essential

negotiation that needs to take place between designers when developing a system; our intent is to provide computational support to that negotiation and to provide a mechanism for capturing the negotiation and its results so that it can be referenced if decisions require revision or if similar decisions are required in related systems.

6.4. DESIGN ALTERNATIVE EVALUATION

One of the key features of the ORCA system is the ability to evaluate the support for the various design alternatives represented in the rationale. This allows the designer to determine the impact on their design choices when requirements, assumptions, and design priorities change. The ability to re-evaluate beliefs (in our case, in the form of alternative evaluations) in the face of changing assumptions is similar to work done by Truth Maintenance Systems (TMSs) (Doyle 1979; de Kleer 1986) although our system stops short of changing the alternative selections by leaving it to the human designers to decide if they agree with the system's evaluation.

The original alternative evaluation performed by SEURAT was a linear function that simply summed up the contributions of the arguments for and against each alternative:

$$AltEvaluation(k) = \sum_{i \in \arg for(k)} amt_i * imp_i - \sum_{j \in \arg against(k)} amt_j * imp_j \qquad (1)$$

This calculation requires that three things be known about each argument: the amount, the importance, and whether the argument is for or against the alternative. The amount specifies how much the alternative conforms to the criteria involved in the argument and is entered by the user when they record the argument. The importance is dependent on the type of argument. It is automatically calculated for arguments concerning requirements, assumptions, alternatives, and other arguments (see Burge 2005 for details). The importance values for each claim are either inherited from the Design Criteria Ontology (allowing a global importance for a criteria to be applied to the design as a whole) or specified at the claim or argument level by the designer.

For ORCA, this evaluation needs to be re-defined to accomplish several goals. First, we wish to allow for uncertainty on the part of the designer when providing their arguments. In addition to an argument's importance, a plausibility or confidence factor for each argument is added as a multiplier to both summation expressions in the above equation. A low confidence factor for an argument would reduce the impact of that argument on the overall assessment. Although this evaluation approach for alternatives still lacks semantic underpinnings, as ORCA development progresses a more sophisticated alternative evaluation mechanism based on decision theory is

to be incorporated in order to determine the best decisions given one's uncertainty and preferences regarding the possible outcomes. Research to develop this alternative evaluation procedure is to investigate work focused on the key problems of assessing the necessary probability and utility functions for a given problem, handling multi-attribute utility functions, and finding useful families of structured probability and utility functions that permit compact representation and efficient inference (von Winterfeldt and Edwards 1986; Wellman 1985). Decision networks (aka influence diagrams) are to be considered as well to aid in the elicitation of probability and utility functions (Howard and Matheson 1984).

8. Conclusions and Future Work

The designers we have consulted with at NASA JPL have made it clear that the ability to capture constraints, assumptions, and collaborative dialog is crucial to the understanding of the designers' intent behind their design and to provide the information needed if the initial design choices need to change in the future. If only the final, or "point design", is documented then the process of re-thinking initial choices is likely to repeat much of the design work that took place earlier. This often happens. This inefficiency would be greatly minimized if the designers working on the changes had access to which alternatives were considered at the initial design stages.

This work will make a number of contributions to research in design computing. Key research results will be a rationale representation designed specifically to support design alternative evaluation during engineering design; algorithms to perform a rigorous evaluation of the alternatives using the rationale with an emphasis on the impact of risk and uncertainty on the decision options; a method to capture collaborative design discussion and incorporate it into the design decision-making process; a demonstration of how design ontologies describing product, process, and criteria can support rationale capture, inferencing, and visualisation; and a prototype system that can be integrated into different design environments to assist in design decision-making.

We have already made progress on several of these objectives. The FAIR-DART conceptual design study (Oberto 2002) described the rationale uncovered during a series of NASA design sessions. We used this actual rationale to provide the requirements needed for our rationale representation and have demonstrated the effectiveness of our representation by capturing key portions of the FAIR-DART rationale in the ORCA system. We have developed an initial Designer Profile Ontology to support collaborative design discussion and extended our initial argumentation representation to allow the capture of rationale from multiple designers within the same

argumentation structure. This development has also been done within our initial prototype system.

We will continue our work on ORCA by continuing the development of the design product, process, and criteria ontologies. We plan to use ontology learning techniques to obtain some of this information from design documentation. We will also enhance the alternative evaluation calculations to use more mathematically sound algorithms for the utility of each choice and to combine the input of multiple designers. Finally, to make this system usable it should be integrated into the design process as much as possible. We plan to evaluate our system within the NASA JPL Team-X environment and expect to receive immediate and quantitative feedback on the validity of our approach. We are seeking opportunities to perform additional 'prototype' applications and we plan to follow that approach with ORCA by integrating it into an actual design environment so that rationale capture and use can become an integral and essential part of the design process.

Acknowledgements

We would like to thank Martin Feather and Leila Meshkat of NASA JPL for providing us with the FAIR-DART study results that motivated this work. We would also like to thank the anonymous reviewers for the Design Cognition and Computing conference for their valuable feedback on our approach.

References

Andreasen, M and Olesen, J: 1990, The concept of dispositions, *Journal of Engineering Design* **1**(1): 17-36.

Andréka, H, Ryan, M and Schobbens, PY: 2002, Operators and laws for combining preference relations, *Journal of Logic and Computation* **12**(1):13-53.

Boehm, B and Bose, P: 1994, A collaborative spiral software process model based on theory W, *in Proc. of the 3rd International Conf. on the Software Process*, IEEE Computer Society Press, CA, pp. 59-68.

Bracewell, R, Ahmed, S and Wallace, K: 2004, DREd and design folders, a way of capturing, storing, and passing on knowledge generated during design projects, *in Proc. of the ASME 2004 Design Automation Conf.* Salt Lake City, USA, pp. 1-22.

Bratthall, L, Johansson, E and Regnel, B: 2000, Is a design rationale vital when predicting change impact? – A controlled experiment on software architecture evolution, *in Proc. of the Int. Conf. on Product Focused Software Process Improvement*, Finland, pp. 126-139.

Brown, DC and Chandrasekaran, B: 1985, Expert systems for a class of mechanical design activity, *in* J.S. Gero (ed.), *Knowledge Engineering in Computer-Aided Design*, North Holland, pp. 259-282.

Brown, DC: 1993, Intelligent Computer Aided Design, *in* A Kent and JG Williams (eds), *Encyclopedia of Computer Science and Technology,* Marcel Dekker, pp. 28.

Burge, J and Brown, DC: 2000, Inferencing over design rationale, *in* JS Gero (ed), *Artificial Intelligence in Design '00*, Kluwer Academic Publishers, Netherlands, pp. 611-629.

Burge, JE and Brown, DC: 2004, An integrated approach for software design checking using rationale, *in* JS Gero (ed), *Design Computing and Cognition '04*, Kluwer Academic Publishers, Netherlands, pp. 557-576.

Burge, JE: 2005, Software Engineering Using design RATionale, PhD Thesis, http://www.wpi.edu/Pubs/ETD/Available/etd-050205-085625/, WPI.

Burge, JE and Brown, DC: 2006, Rationale-based support for software maintenance, *in* A Dutoit, R McCall, I Mistrik, B Paech (eds), *Rationale Management in Software Engineering*, Springer *(to appear)*.

Cantwell, J: 1998, Resolving conflicting information, *Journal of Logic, Language, and Information* **7**: 191-220.

Chandrasekaran B, Goel AK and Iwasaki Y: 1993, Functional representation as design rationale, *Computer* pp. 48-56.

Conklin J and Yakemovic KB: 1991, A process-oriented approach to design rationale, *Human-Computer Interaction* **6**(3-4): 357-393.

Conklin, J and Burgess-Yakemovic, K: 1995, A process-oriented approach to design rationale, *in* T Moran and J Carroll (eds), *Design Rationale Concepts, Techniques, and Use*, Lawrence Erlbaum Associates, Mahwah, NJ, pp. 293-428.

Conklin, J: 1996, Capturing organizational memory, *GDSS Working Paper*, http://www.touchstone.com/tr/wp/CapOrgMem.html

Conklin, J, Selvin, A, Buckingham Shum, S and Sierhuis, M: 2003, Keynote address, *in* H Weigan, G Goldkuhl, A de Moor (eds), *Proc. of the Language-Action Perspective on Communication Modelling*, Tilburg, The Netherlands.

Conklin, J, Selvin, A, Buckingham Shum, S and Sierhuis, M: 2001, Facilitated hypertext for collective sensemaking, 15 years on from gIBIS, *Technical Report,* Knowledge Media Institute, KMI-TR-112.

de Kleer, J: 1986, An assumption-based truth maintenance system, *Artificial Intelligence* **28**(2): 127-162.

Doyle, J: 1979, A truth maintenance system, *Artificial Intelligence* **12**(3): 231-272.

Eris O, Hansen PHK, Mabogunje A and Leifer L: 1999, Toward a pragmatic ontology for product development projects in small teams, *Proceedings of International Conference on Engineering Design,* Technische Universität München, Munich, pp. 1645-1950.

Fowler, DW, Sleeman, D, Wills, G, Lyon, T and Knott, D: 2004, The designers' workbench: Using ontologies and constraints for configuration, *Proc. of The 24th SGAI International Conf. on Innovative Techniques and Applications of AI*, Cambridge, UK, pp. 209-221.

Garcia, A, Howard, H and Stefik, M: 1993, *Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design*, TR 82, Stanford Univ. Center for Integrated Facility Engineering, Stanford, CA.

Gero, JS: 1990, Design prototypes: A knowledge representation schema for design**,** *AI Magazine* **11**(4): 26-36.

Gruber, T:1990, *Model-Based Explanation of Design Rationale*, Technical Report KSL 90-33, Knowledge Systems Laboratory, Stanford University.

Howard, RA and Matheson, JE: 1984, Influence diagrams, *Principles and Applications of Decision Analysis* **2**:690-718.

Hubka, V and Eder, WE: 1996, *Design Science*, Springer-Verlang, London.

Japikse, R, Langdon, PM and Wallace, KM: 2003, Structuring engineering design information using a model of how engineers' intuitively structure design information, *Proc. of the, 14th International Conf. on Engineering Design, Sweden*, pp. 433-434.

Karacapilidis, N and Papadias, D: 2001, Computer supported argumentation and collaborative decision making: The HERMES system, *Information Systems* **26**(4): 259-277.

Karsenty, L: 1996, An empirical evaluation of design rationale documents, *Proc.of the Conf. on Human Factors in Computing Systems*, Vancouver, BC, pp. 150-156.

King, JMP and Bañares-Alcantara, R: 1997, Extending the scope and use of design rationale records, *AIEDAM* **11**(2):155-167.

Kitamura, Y, Kashiwase, M, Fuse, M and Mizoguchi R: 2004, Deployment of an ontological framework of functional design knowledge, *Advanced Engineering Informatics* **18** (2): 115-127.

Klein, M: 1997, An exception handling approach to enhancing consistency, completeness and correctness in collaborative requirements capture, *Concurrent Engineering Research and Applications* **5**(1): 37-46.

Lee, J: 1997, Design rationale systems: understanding the issues, *IEEE Expert* **12**(3): 78-85.

Lin, J, Fox, MS and Bilgic, TA: 1996, Requirement ontology for engineering design, *Concurrent Engineering: Research and Applications* **4**(3): 279-291.

MacLean, A, Young, RM, Bellotti, V and Moran, TP: 1995, Questions, options and criteria: elements of design space analysis, *in* T Moran and J Carroll (eds), *Design Rationale Concepts, Techniques, and Use*, Lawrence Erlbaum Associates, NJ, pp. 201-251.

Matzkevich, I and Abramson, B: 1992, The topological fusion of Bayes nets, *in Proc. UAI'92*, pp. 191-198.

Maynard-Reid II, P and Chajewska, U: 2001, Aggregating learned probabilistic beliefs, *in Proc. of the 17th Conf. on Uncertainty in Artificial Intelligence (UAI'01)*, pp. 354-361.

Maynard-Reid II, P and Shoham, Y: 2001, Belief fusion: Aggregating pedigreed belief states, *Journal of Logic, Language, and Information* **10**(2): 183-209.

Maynard-Zhang, P and Lehmann, D: 2003, Representing and aggregating conflicting beliefs. *Journal of Artificial Intelligence (JAIR)* **19**: 155-203.

Oberto, R: 2002, *FAIR/DART Option #2*, Advanced Projects Design Team, NASA Jet Propulsion Laboratory.

Peña-Mora, F, Sriram, D and Logcher, R: 1995, Design rationale for computer-supported conflict mitigation, *ASCE Journal of Computing in Civil Engineering* **9**(1): 57-72.

Pennock, DM, Maynard-Reid II, P, Giles, CL and Horvitz, EA: 2000, Normative examination of ensemble learning algorithms, *in Proc. ICML'00*, pp. 735-742.

Sim, S and Duffy, A: 1994, A new perspective to design intent and design rationale, *in Artificial Intelligence in Design Workshop Notes for Representing and Using Design Rationale*, pp. 4-12.

Taura, T and Kubota, A: 1999, A study on engineering history base, *in Research in Engineering Design* **11**(1): 45-54.

Tong, C and Sriram, D: 1992, Introduction, *in* C Tong and D Sriram (eds) *Artificial Intelligence in Engineering Design*, Volume 1, pp. 1-53.

Wellman, M: 1985, *Reasoning about Preference Models*, Technical Report MIT/LCS/TR-340, Laboratory for Computer Science, MIT.

von Winterfeldt, D and Edwards, W: 1986, *Decision Analysis and Behavioral Research*, Cambridge University Press, Cambridge.

FROM ARCHITECTURAL SKETCHES TO FEASIBLE STRUCTURAL SYSTEMS

RODRIGO MORA, HUGUES RIVARD
ETS, Canada

and

ROLAND JUCHMES, PIERRE LECLERCQ
University of Liège, Belgium

Abstract. The goal of this research is to propose an integrated approach to incorporate structural engineering concerns into architectural schematic designs for timely and well-informed decision making. This is done through a platform that is based on two software prototypes, EsQUIsE for capturing and interpreting architectural sketches, and StAr for assisting engineers during conceptual structural design. An integrated information model is provided for communication. Given the dissimilar "quality" of the information managed by both prototypes, sketch interpretation mechanisms are also required to "tune-up" communications for bringing the sketch to a precise structural engineering definition. As a result, the engineer can propose feasible structural systems earlier than usual.

1. Introduction

Conceptual design is explorative in nature. During this stage, designers refine design ideas and explore design alternatives under uncertainty (i.e. based on still limited and assumed information). Each alternative requires minimal resource commitment as it can be discarded easily and start a new, or evolve (through changes and/or refinements) to become a conceptual design solution. Accuracy is minimized for efficiency in alternative generation, and detail is minimized for effectiveness in design intent representation. During conceptual building design the most salient characteristics of the building are defined. Thus, major decisions are made regarding the building architecture, such as the internal configuration of spaces and physical elements that give shape to the building form, as well as major aspects of supporting engineering systems, such as materials, type, layout and initial dimensions.

J.S. Gero (ed.), Design Computing and Cognition '06, 675–694.
© 2006 *Springer. Printed in the Netherlands.*

There is a consensus that engineering feedback is required by architects as early as possible during building design. Architects and engineers are complementary design actors with different expertise, needs and priorities, as well as different working timeframes and design tools. Architects use sketches during conceptual design for exploration and development of their own ideas, and for communicating them (Meniru et al. 2003). Engineers must accommodate to the architect's work pace, as well as her/his evolving design representations (Mora et al. 2006a). In the absence of sketches, engineering feedback can be provided to the architect from experience, based on overall building parameters, such as building type, size, number of floors, etc. Only when architectural sketches are made available to the engineer, can he/she uncover potential structural problems in the architectural design and devise and compare structural load transfer solutions that integrate well to the architecture. However, there is no computer tool available to assist engineers in proposing feasible structural systems from architectural sketches.

Nowadays, advanced computer modeling tools are available to support structural system generation and the integration to the architecture (Khemlani 2005). This kind of support is model-based since it relies on the geometric and data modeling capabilities of a building information model (BIM) that combines the building architecture with other disciplines. For example, Revit Structure by Autodesk enables the creation of a structural model directly from an architectural model created using Revit Building (Autodesk 2005). Similarly, the IdeCAD platform (IdeCAD 2005) combines IdeCAD Architectural and IdeCAD Structural applications. Those applications however constrain architects and engineers to share a proprietary model-based platform from one software vendor.

Software interoperability (i.e. information exchange) has been devised as an alternative to proprietary BIM. The industry foundation classes (IFCs) developed by the International Alliance for Interoperability (IAI) have become the standard for information exchange among building related applications (IAI 2005). The IFCs use a standard object-based representation model (IFC model) with object descriptions encompassing the different domains involved in building design, fabrication, construction and operation. Considering structural design, four IFC extension projects have been completed in the structural engineering domain: steel frame constructions (ST-1), reinforced concrete structures and foundation structures (ST-2), pre-cast concrete construction (ST-3), structural analysis model and steel constructions (ST-4). For the IFC ST-1 project, a liaison has been made between the IAI and the developers of CIMsteel Integration Standard version 2.0 - CIS/2 (CIMsteel 2005). CIS/2 provides a semantically explicit representation for describing steel structures.

The above proprietary and standard models constitute a significant step forward towards building integration and collaboration among practitioners. However, these models have not been developed with the aim at supporting conceptual design. Recent efforts by the IAI have aimed at extending the IFC applicability towards earliest design stages (IAI 2005). However, this extension project focuses on architectural programming and space layout planning and not on integrated design.

The goal of this research project is to enable early integrated design between architects and structural engineers through computer support for enabling structural engineering concerns to be considered in the architect's schematic design explorations without interfering with his/her design workflow. This is a joint research between the LuciD group from the University of Liège, in Belgium and the team of the Canada Research Chair in Computer-Aided Engineering for Sustainable Building Design at ETS in Montreal. It is carried out in three stages. In a first stage timely and well-informed communication and decision-making are supported. In a second stage negotiation and changes will be supported for conceptual design evolution. In a third stage real-time collaboration will be supported.

To enable early integrated design, a collaboration platform is envisioned to assist architects and structural engineers. The platform is based on two software prototypes, EsQUIsE developed by the LuciD group (LuciD 2005) for capturing and interpreting architectural sketches, and StAr developed at ETS for assisting engineers during conceptual structural design. An integrated information model is provided for communication. Given the dissimilar "quality" of the information managed by both prototypes, sketch interpretation mechanisms are also required to "tune-up" communications for bringing the sketch to a precise structural engineering definition.

This paper presents the results from the first stage of this joint research project. The paper is organized as follows: the second section presents previous work in supporting engineering from sketches. Section three introduces the existing complementary early design support systems. Section four describes the information exchange model between these systems. Section five describes the main factors to be considered in creating conceptual structural systems from imprecise sketches. Section six describes the steps followed for transforming imprecise sketches into feasible structural solutions. An example of support for structural design from sketches is finally presented and evaluated.

2. Previous Work in Supporting Engineering from Sketches

Several researchers have worked on making the computer play an active role during sketching by recognizing the different components (i.e. shapes, symbols, relationships and text) of a sketch in real time and assigned them

roles during design. In this way, formal component and system representations are built from sketches. Once a formalized representation is obtained from a sketch, it can be used for simulating the behaviour of the component or system it describes.

In the field of mechanical engineering, the prototype ASSIST (Alvarado and Davis 2001) helps designers to sketch simple mechanical devices in plan or section. Then, it constructs a formalized representation from the sketch and uses this representation to simulate the behaviour of the mechanical device in a two-dimensional kinematic simulator. Lipson and Shpitalni (2000) developed a geometric correlation method for transforming 2D isometric sketched views of mechanical parts into 3D solids that are analyzed for viability and manufacturability.

In the field of architecture, there are three notable research prototypes that enable the sketch to play an active role during design. SKETCH (Zelenik et al. 1996) uses gesture-based recognition methods to construct 3D scenes from 2D perspectives and isometric views. The Electronic Cocktail Napkin (Gross and Do 1996) employs context-based recognition to identify configurations from schematic drawings and diagrams in two-dimensions. The formal representation it builds from sketches is used for constraint-based editing and enables the tool to be used as a front-end for interactive simulations. EsQUIsE (Leclercq 1999) has been developed for capturing and interpreting the architect's sketch by recognizing architectural concepts such as walls, functional spaces and space topology. The research described in this paper relies on EsQUIsE as the architectural tool, as described in more detail in the next section.

To the authors' knowledge no research has been carried out to date that relies on free hand sketches for structural design purposes. Miles et al. (2004) have proposed using automated constraint checking for restricting the search for structural solutions from sketches. However, the sketches used are simplified two-dimensional drawings rather than actual free hand sketches (free hand sketching is used only for annotations). The process of bringing free-hand sketches to structural precision involves complexities that have not been contemplated so far by sketch recognition tools. Mechanical assemblies are constructed bottom-up from individual pieces that are related together mainly through mating relationships, thus favoring a bottom-up sketch recognition process. By contrast, structural system solutions are geometrically and topologically related to the building architecture's forms and functions in several ways.

A main premise of this research is that computers cannot and should not automatically transform imprecise architectural sketches into precise representations to be used for structural design. Some architectural concepts and relationships are more easily recognizable by the trained eye of an engineer than by a computer and therefore it is in the engineer's

responsibility to identify them. Here, it is proposed that the process of making a sketch more precise should be carried out by the computer with implicit guidance from both the architect and the engineer.

3. The Existing Complementary Early Design Support Systems

EsQUIsE and StAr are complimentary software prototypes that support early building design. A platform based on these prototypes permits architects and engineers to work using tools and concepts that are just suited to their particular design needs. The prototypes have been adjusted to enable smooth two-way communications between them.

EsQUIsE captures and interprets architectural sketches in real time. The designer creates his/her drawings with an electronic pen. The keyboard is never used and no menu is needed to depict the building. The EsQUIsE interface is designed to be as close as possible to the architect's traditional and natural way of work. Thanks to its Multi-Agent System, EsQUIsE is able to classify and recognize the different elements composing the sketch: dotted lines, hatchings, architectural symbols etc. Moreover, EsQUIsE disposes of a hand written recognition module, which allows the architect to give complementary annotations about the building, i.e. the rooms' names or the walls' compositions. Once the drawing elements are recognized, EsQUIsE builds a 3D model of the building and, assisted by its implicit architectural knowledge base, it can complete the characteristics not explicitly given by the designer. For example, as wall compositions are rarely specified in the preliminary design, EsQUIsE can automatically select relevant compositions according to the spaces separated by the walls. The 3D Model can then be used by various simulation tools such as: real time walk-throughs, assessment of yearly buildings energy needs, and building and functioning costs. These, provide the designer with qualitative and quantitative information on the performances of the designed product.

StAr is a prototype system that assists engineers in the inspection of a 3D architectural model (e.g. while searching for continuous load paths to the ground) and the configuration of structural solutions. Assistance is based on geometrical reasoning algorithms (Mora et al. 2006b) and an integrated architecture-structure representation model (Mora et al. 2006a). The algorithms which are enhanced with implicit generalized structural/architectural knowledge assist the engineer in reasoning from a 3D model of the building architecture and the structural system for synthesizing structural solutions. They use the geometry and topology of the design model to construct new geometry and topology, and to verify the model. Work is currently in progress to provide StAr with a graphical user interface

(GUI) for inputs to replace the current interface with alphanumeric interactions with graphical outputs. A conceptual design knowledge module is also being developed to complement the existing algorithms with explicit structural engineering knowledge.

In this research, the StAr prototype was enhanced with new capabilities for accepting and processing three-dimensional models produced by EsQUIsE from architectural sketches. These capabilities are built upon the theory in described Section 5 and are explained in Section 6.2. An initial information exchange model was also developed to make these prototypes communicate. This model is described in the next section.

4. Information Exchange Model

The integration between EsQUIsE and StAr is based on the principle that even though architects and structural engineers use their own domain concepts and individual design tools, they produce a single and unique building design solution that should integrate concerns from both parties. A common building model is therefore being developed that incorporates early architectural and structural concerns. This model evolves as is augmented by the architect and the engineer, each adding and using relevant information and ignoring irrelevant one. Thus, information that may not be critical to one actor at some point can become critical later on in the process. The need for a tailored-made integrated model comes from the fact that the IFC model does not respond to key requirements of conceptual structural design, namely: to minimize accuracy for efficiency in alternative generation, to minimize detail for effectiveness in design intent representation, and to facilitate multiple design iterations between architects and engineers for design exploration and evolution.

As a starting point in this research, a simplified information-exchange model has been developed that is initially used to store an architectural 3D model produced from sketches by EsQUIsE, Figure 1. It is static, since it supports only one design iteration between the architect and the engineer. In this model, the architecture is stored in two sub-models: a graphic model and a building model. The graphic model contains the following entities: transparencies, colors, lines, limits, contours, and text (i.e. for annotations). The building model contains the following entities: stories, spaces, and partitions. Spaces are in turn sub-divided into elementary spaces (spanning one storey) and partitions into elementary partitions that can be vertical (i.e. walls or columns) and horizontal (i.e. slabs). Partitions may have openings (i.e. doors and windows). All architectural entities have a basic description including a type for identification. Note that this information is interpreted by EsQUIsE and never explicitly entered by the architect. Except for the space names, since the wall types and openings are inferred by EsQUIsE based on the location (interior/exterior) and on the spaces enclosed. This

information is sufficient for transmitting architectural design intent to the structural engineer. Then, after a structural solution is generated by StAr it is appended at the XML file as a graphic model of the structure and is visualized by the architect in EsQUIsE.

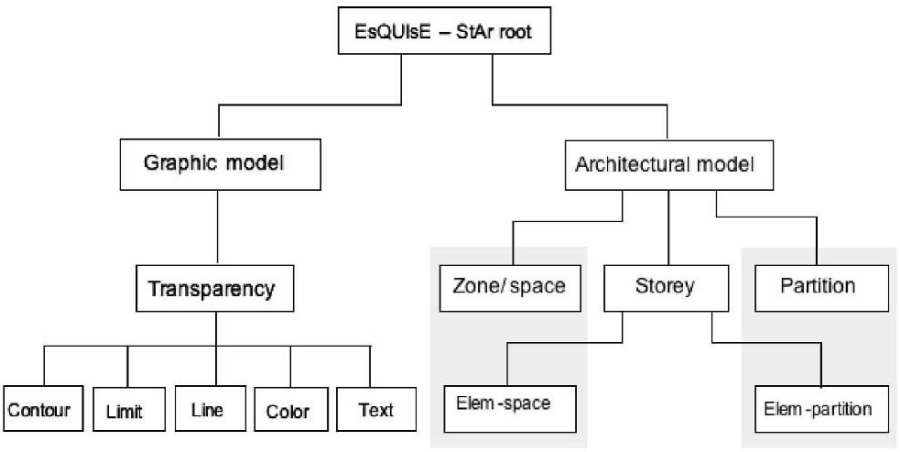


Figure 1. Information exchange model.

5. Factors to Consider for Dealing with Sketch Imprecision

The lack of precision of sketches complicates the engineer's task of finding continuous load paths to the ground (because architectural patterns are not so evident from a sketch) and proposing structural solutions (because some dimensional accuracy is required to make decisions). Before proposing structural solutions from a sketch, the engineer has to uncover intended patterns in the layout of walls and columns. The effectiveness of this task depends on the geometric complexity of the building architecture.

5.1. INTENDED ARCHITECTURAL PATTERNS

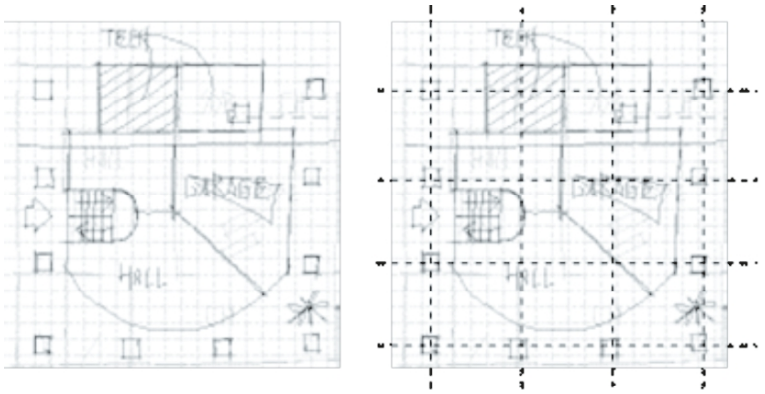
In the search for continuous load paths to the ground from sketches, an engineer performs the following tasks (adapted from Lecomte 2003):

1. Vertically-wise: recognize intended vertical continuity of walls and columns (if placed by the architect).
2. Horizontally-wise: Identify alignments of walls and columns that may be intended to be collinear. Identify parallel, orthogonal and other layouts of wall and column lines.
3. Dimensionally-wise: identify intended space dimensions defined by enclosing walls and columns.

Once the patterns inferred by walls and columns have been recognized, they are used by the engineer to define structural grids that determine the primary layout of the structural system.

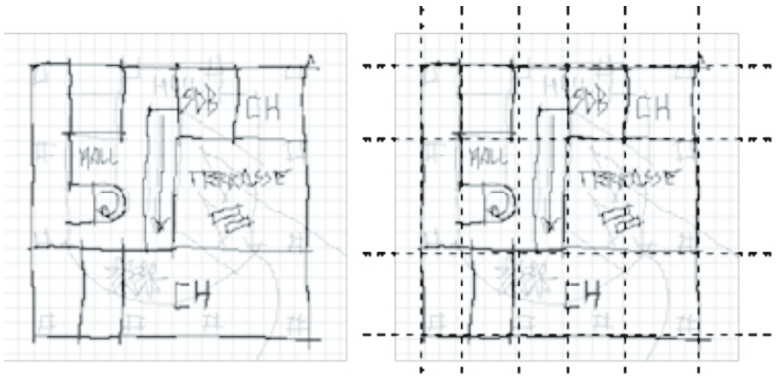
Figures 2 and 3 illustrate the task of inferring structural grids from sketches. The figures show a sketch on the left and the same sketch with a superimposed grid on the right. Figure 2(b) shows that columns are not strictly aligned in the sketch. In addition, it shows that there is a somehow squared grid made by the columns. Figure 3(b) shows that walls do not match the patterns made by the grids. It also shows no uniformity in the gridlines that may be intentional.

This pattern recognition task may require some clarifications on the part of the architect, as to what is intended and what is not. For example, in Figure 3(b) dimensions may be easily adjusted so that pattern uniformity is achieved. Achieving uniformity is preferred by the engineer (i.e. it means equally sized structural elements) but not necessarily by the architect.



(a) sketch (b) inferred grid

Figure 2. First example of a sketch and some inferred grid lines.



(a) sketch (b) inferred grid

Figure 3. Second example of a sketch and some inferred grid lines.

In addition, in defining structural grids, the engineer uses dimensions approximated to the precisions used in construction practices (e.g. use "5.5 m" instead of "5.6567 m"). These grid dimensions are likely dependent on some characteristic functional dimensions as defined by the architect. However, since sketches are not precise, the engineer has to obtain intended dimensions from the architect or infer them.

5.2. GEOMETRIC COMPLEXITY OF THE ARCHITECTURE

For the engineer, the effectiveness in finding patterns in the architectural design and making the sketch more precise depends on its geometric complexity. Figure 4 uses set theory to elaborate on this point, where: set "G" represents the patterns formed by implicit or explicit project grids, set "A" represents the patterns formed by architectural walls and columns, and set "S" represents the patterns formed by the vertical structural elements.

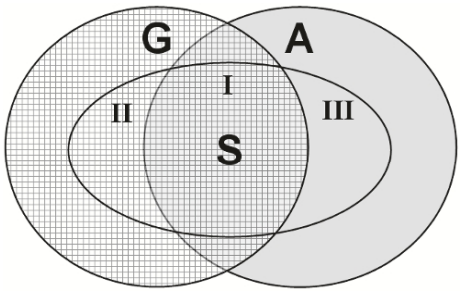


Figure 4. Vertical grids, patterns and architectural complexity.

In the search for continuous load paths to the ground, the engineer seeks to lay out structural supports in patterns (i.e. structural grids) that match wall and column patterns implicit in the architecture, unless particular architectural conditions exist that force the structure to fall outside these patterns. Set "S" is divided in three zones. In zone I, structural patterns are fully contained within the architectural patterns, so that all structural members are either hidden within architectural elements or intentionally visible, with no unwanted structural members lying in unwanted locations. Note that structural elements that are intentionally visible are also architectural elements since they are part of the architectural concept. In zone II, structural elements still adjust to the common patterns between the architecture and the structure. However, due to structural dimensional constraints the engineer must place columns inside spaces, i.e. outside the set of space establishing elements. In zone III, some architectural elements that fall outside the common patterns are structurally relevant. The engineer may extend the structural grids and provide a special local framing lying outside

the common pattern to integrate such architectural elements to the structural system.

For small and simple buildings it is expected that most vertical structural elements fall in zone I. For buildings with large spaces and constrained structural dimensions it is expected that more elements will fall in zone II. The more complex the building geometry, the more structural elements are expected to fall in zone III. Dealing with sketch imprecision is simpler if structural patterns fall in zones I and II because these patterns correspond to architectural ones (i.e. within common grids). Buildings requiring structural patterns to fall in zone III introduce difficulties in making the sketch more precise. In such cases, the engineer may require clarifications by the architect before proposing structural supports.

6. Bringing an Architectural Sketch to Structural Engineering Precision

The process of bringing an architectural sketch to structural engineering precision is carried out in two stages: (1) bottom-up automatic sketch interpretation in EsQUIsE, and (2) top-down interactive architectural-structural integration in StAr, Figure 5. In both prototypes, the designer implicitly leads the sketch interpretation and refinement process based on his/her intrinsic design actions.

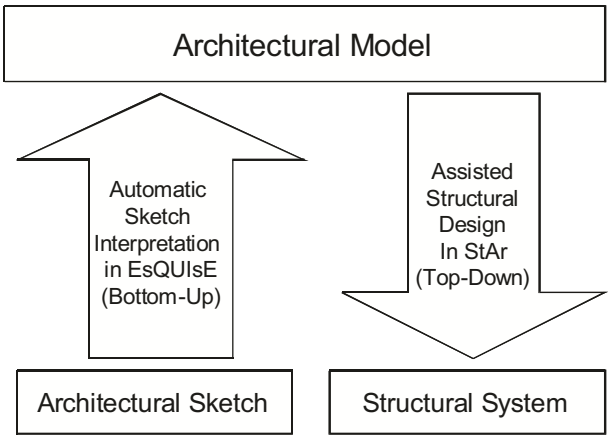


Figure 5. Process of bringing the sketch to structural engineering precision.

6.1. DEALING WITH SKETCH IMPRECISION IN EsQUIsE

Sketches are a means of heuristic exploration that help designers discover design interpretations opening the door to new solution spaces. Imprecision and ambiguity are essential for sketching to avoid fixing design solutions too soon. In imprecise sketches the orientation of strokes varies slightly from sketch to sketch, which permits re-directing the design towards new and

unexpected solutions. This phenomenon largely studied, called "lateral transformation" (Goel 1995) or "seeing as" (Goldshmidt 1991), justifies the fact that an application that supports design from sketches should not attempt to "correct" the strokes but rather maintain this ambiguity necessary for creativity. EsQUIsE relies on this principle to deal with sketch imprecision.

The process of sketch interpretation in EsQUIsE is a bottom-up linear process with graphic strokes by the architect as inputs and producing a sketch interpretation as output. Such interpretation involves not only the recognition of meaningful entities in the sketches but also their refinement and formalization into architectural models, while leaving the original sketches unaltered. The process is carried out in three main stages:

1. Low level data preparation for subsequent stages. At this stage the noise in the data is reduced and the strokes are replaced by curves: straight lines, arcs, etc. At this level, the intrinsic stroke imprecision is first treated, independent of any context. This can be considered as random noise (Sezgin and Davis 2004) caused by the digitizing surface and the imprecision of the designer. Next, a time context when the stroke is drawn is used to further overcome imprecision. Emphasizing a stroke for example alters its layout.

2. A classification stage in which line segments are grouped into more complex graphic entities: hatchings, symbols, legends, etc. This is followed by recognition of the graphic entities in the drawing. At this stage, imprecision is treated at the level of symbol classifiers that are used for graphic recognition. These symbols must tolerate some variability in a drawing since two symbols are never identical.

3. An interpretation stage in which a building model is built that corresponds to the original sketch. Imprecision at this level is concerned with the semantic of the objects being represented by the sketch. For example, the distance between two strokes "supposedly adjacent" can be eliminated as the system recognizes that these represent a single wall.

Thus, EsQUIsE removes excess information and noise from a sketch, straightens line segments, and closes space perimeters (where applicable) to produce a complete and unambiguous 3D model of the building. Nevertheless, the 3D model that is transferred from EsQUIsE to StAr is still imprecise since no correction of alignments, parallelisms, etc. is realized. This is more naturally done by the trained eye of an engineer using StAr.

6.2. DEALING WITH SKETCH IMPRECISION IN StAr

Bringing an architectural sketch to structural engineering precision is done semi-transparently by StAr with minimum interference with the engineer's conceptual design workflow.

Figure 6 illustrates a top-down approach for conceptual structural design that is used by StAr. In the figure, the process is carried out linearly, with backtracking, through four main activities (courier bold typeface indicates model entities). After inspecting the building architecture looking for load paths to the ground, the engineer proposes overall load transfer solutions, while considering applied loads. These solutions are described in terms of supporting structural assemblies and corresponding material(s) and worked out based on tentative structural grids. Structural grids determine the layout of the primary structure including structural bays, vertical support locations (at gridline intersections), and floor spans. Once an overall load transfer solution is selected, structural assemblies are specified and structural elements are laid out and connected together.

StAr uses structural grids defined by the engineer for dealing with sketch imprecision. Depending on the scale of the sketch, scale-based tolerance factors are defined and used to project architectural elements that play a structural role into grids. The efficiency of this process depends on the geometric complexity of the architectural design (Section 5.2), as well as the following factors: the approximate scale of the sketch, the free hand sketching precision of the architect, and the semantic and dimensional hints that the architect may want to include in the design. The process for dealing with sketch imprecision is as follows.

From an imprecise three-dimensional model derived from sketches the engineer carries out activities number 1, 2 and 3 in Figure 6 just like with a well-defined three dimensional model: the engineer proposes overall load transfer solutions with reference gridlines, groups spaces to define structural zones, and positions structural assemblies using the gridlines and the floor levels as reference. Then, StAr generates the physical structural elements as indicated in activity number 4 in Figure 6.

During activities number 3 and 4, StAr detects if the structural system accurately corresponds to the architectural design using geometric algorithms with tolerance factors. During activity number 3, before generating the physical structure, StAr verifies that shear walls and bracings defined by the engineer based on gridlines are housed by (imprecise) architectural walls. During activity number 4, StAr generates structural elements within the structural grids specified by the engineer. In doing so, StAr checks for architecture-structure correspondence by attempting to project architectural elements into the structural grids. Activity number 4 is carried out in two stages: in a first stage vertical elements are positioned, and

in a second stage the shape of the slabs is determined by taking into consideration the vertical correspondence of slab shapes between stories.

When, structural geometries fall outside the structural grids (zone III in Figure 3), attempts are made to project their geometry into a grid direction. If a projection fails, then the original direction is left unchanged as it may have been skewed intentionally by the architect. Therefore, the approach accepts oblique walls and slab perimeters in plan with respect to the grids.

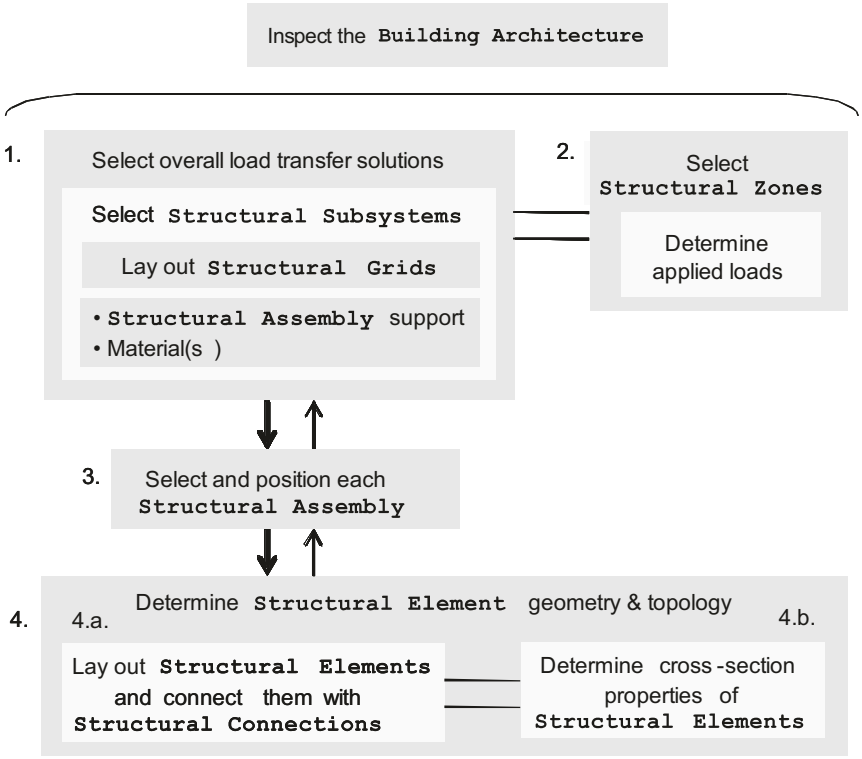


Figure 6. Top-down conceptual structural design.

Missing supports

After an initial structure is generated by StAr it is likely that unsupported elements or un-integrated elements will be found either because they fall in zone III, Figure 4, and/or because the sketch model is not clear enough for recognition by StAr. Therefore, StAr provides two ways for specifying supporting local frames: It lets the engineer extend the structural grids by placing additional reference gridlines. These gridlines will provide new alignments for additional vertical supports. It incorporates implicit knowledge that is used to search for permanent supporting walls from the architecture that may have not been detected by the engineer.

7. Example of Using Architectural Sketches for Structural Design

A simple example is used to illustrate the collaboration approach currently implemented. The building is a two-floor firehouse with parking space for fire trucks on the ground floor, offices on the floor above and a terrace on the roof. The schematic architectural design is performed by the architect in plan using EsQUIsE, as illustrated in Figures 7 and 8. Also, as shown in these Figures, the architect has tentatively placed some interior columns.

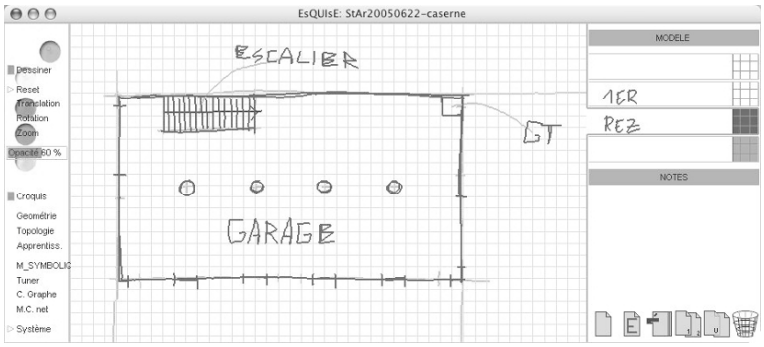


Figure 7. Ground floor of the firehouse in EsQUIsE.

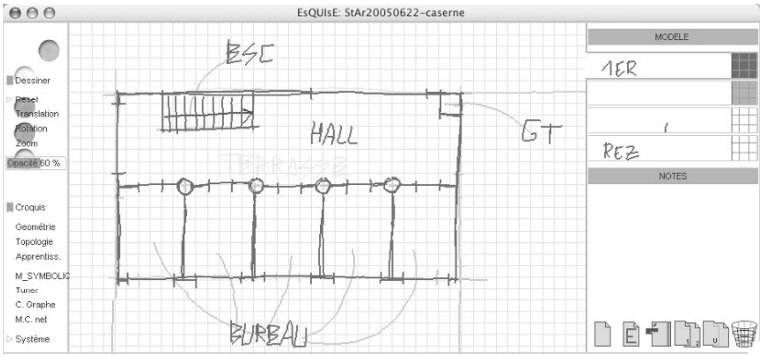


Figure 8. Second floor of the firehouse in EsQUIsE.

As the architect performs schematic design in floor plan, EsQUIsE recognizes walls and spaces and synthesizes a 3D model shown in Figure 9.

Once the architect is satisfied with the design it is then exported in an XML file. The file is read by StAr and the architecture is visualized and inspected by the engineer. From the file, the engineer has access not only to geometric information but also to project information such as occupancy types, areas, spaces and wall types. Next, the engineer inspects the architectural model. Since the geometry is simple and the size of the building is small, no potential structural problems are expected within the architecture. No visible constraints are apparent either.

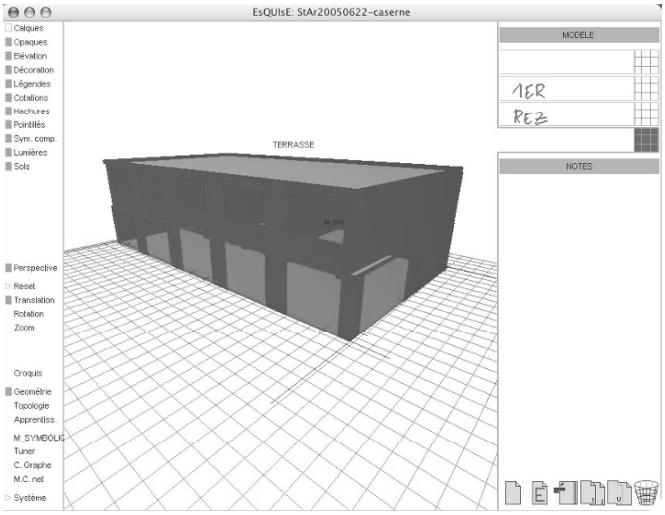


Figure 9. 3D model in EsQUIsE.

The engineer recognizes implicit project grids from the configuration of walls and columns, measures approximate distances and identifies walls and columns that can be used as vertical supports. Next, the engineer selects a load transfer solution by specifying structural grids, Figure 10, selecting concrete as structural material, and selecting structural subsystems as follows: for lateral support use rigid frames with shear walls, and for gravity support use flat slabs poured in place. Resulting floor spans are short and within typical limits based on practice. The engineer extends the grids to provide support for the staircase. Since the architectural patterns are unambiguous, no clarifications are required from the architect.

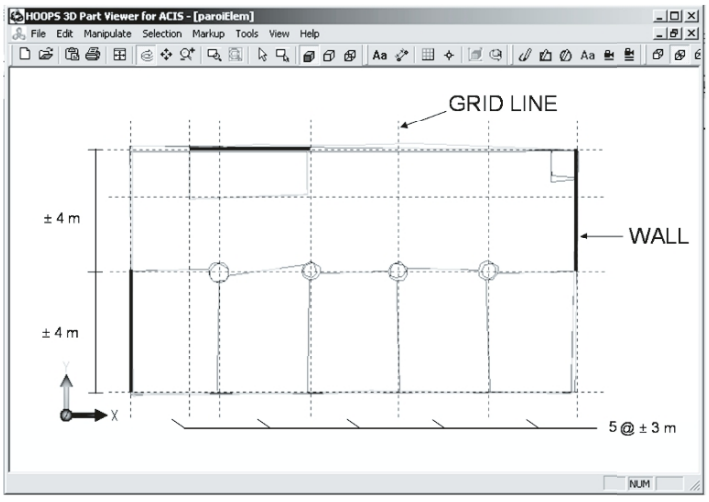


Figure 10. Structural grids.

In Figure 10, the engineer has placed additional gridlines to lay out supports for the staircase. In the event that the engineer does not place these gridlines, StAr searches for supports in the stories below (see "missing supports" in Section 6.2). In this case, an interior wall in the "x" direction is found that could be used for support. However, StAr notices that the wall has an opening that spans almost the entire wall length. Therefore, StAr places two columns in its corners and a beam on top supported by them.

Next, the engineer defines structural zones by grouping spaces. StAr computes the applied load for each zone depending on the function of the spaces grouped. In this case three structural zones are defined one for the ground floor, another for the second floor, and another zone for the terrace. The engineer verifies with the architect that there are no constraints in spaces such as maximum floor depths and building heights. However, it is passed unnoticed that the depth of the slabs is constrained by the 200 mm depth of the lintels in the façade.

Then, the engineer positions and specifies structural assemblies either individually or by group. Frame assemblies are positioned at gridline locations and floor assemblies are selected from architectural slabs. At this point, the engineer selects the architectural walls to become wall stacks and determines their length as bounded by gridlines. StAr automatically checks that these wall stacks are housed by architectural walls. If not, StAr warns the engineer while presenting him/her with a maximum-width continuous wall strip. Then, StAr computes accurate wall geometries which are shown in Figure 10 as bold lines.

After structural assemblies are specified, StAr verifies that the floor spans fall within specified thresholds given elsewhere (e.g. Schodek 2004) and generates the physical structure, Figure 11.

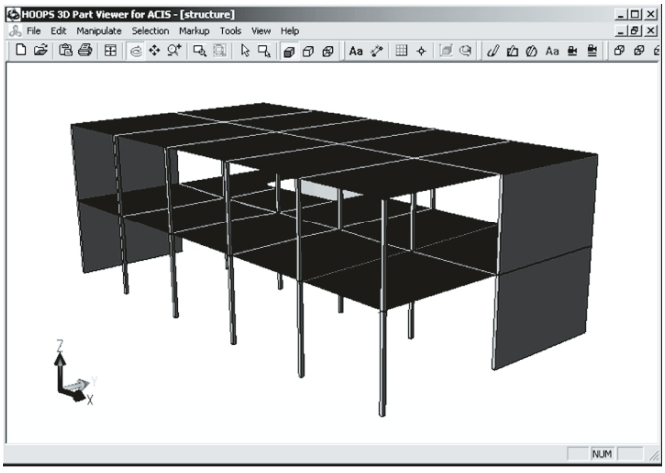


Figure 11. Structural system generated by StAr.

Structural elements are dimensioned initially by StAr using span/depth ratios for beams and slabs, and thickness/height ratios for columns (e.g. Schodek 2004). StAr verifies that no dimensional constraints are violated from the architecture. In this case, span depth ratios produce a slab depth of 150 mm, which is smaller than the lintel depth of 200 mm (as verified by StAr). In the current version, tributary areas and imposed loads are not considered by StAr in the initial dimensioning. This capability will be provided with a conceptual design knowledge module (Section 3).

If the engineer wants to lay out another structural alternative, he/she can start again from the initial architectural file and repeat the process described above. In this example the engineer realizes that interior columns are not necessary. Then, instead of having a layout with two structural bays of four meters, Figures 10 and 11, in one direction, one structural bay of eight meters is proposed. Next, the engineer defines the structural grids accordingly and selects the gravity and lateral load transfer subsystems. For lateral support rigid frames and shear walls are used again, while slab and beams poured in place are used for both slabs to resist the gravity loads. The resulting alternate structural system layout is shown in Figure 12. This layout produces deeper beams and wider exterior columns, which as verified by StAr, pose no problem according to the architectural requirements since perimeter beams are shallower than the requirement (maximum 200 mm). From the engineering stand point, the second structural solution is more convenient because it reduces the amount of material used, and the load on the foundations. For the architect, the second alternative is also more convenient because it provides more space flexibility.

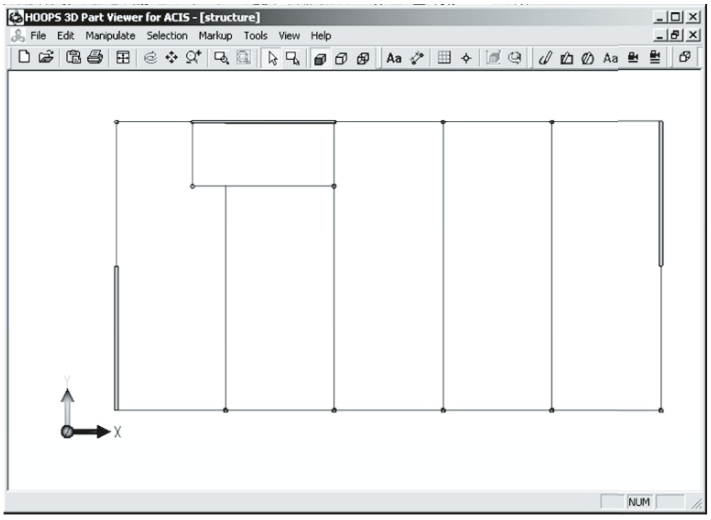


Figure 12. Floor framing layout for the first floor (second alternative) in StAr.

Finally the structural alternative is sent back to EsQUIsE where it can be visualized by the architect. After visual inspection of the structure, the architect may decide to modify the sketch (i.e. to modify the layout and dimensions of the offices in the top floor) or develop another alternative.

8. Evaluation of the Approach

Extracting meaningful architectural information from imprecise sketches and making it suitable for structural engineering is more difficult than from detailed and well-formed architectural models. Thus, the value of this research lies in demonstrating how computers can help engineers to uncover architectural intents implicit in sketches, make sketches more precise, integrate structural solutions to early architectural designs, make informed decisions, and provide timely engineering feedback to the architect. This has been partially achieved with the example.

Architectural sketches incorporate meaningful information that is used by StAr to achieve architecture-structure integration as follows:

- The computer checks that precise structural walls are completely housed by permanent, imprecise architectural walls.
- Space and wall functions are considered during the search for vertical supports. In the example, if the engineer does not place additional gridlines for laying out interior supports for the staircase, StAr locates these supports taking into account space and wall functions.
- Imprecise space geometries and space functions are used for generating the precise geometry of slabs and their openings (i.e. shafts and staircases) and for load determination.
- Architectural constraints (implicit in the sketch) are checked for the positioning of columns, and for beam/floor depths.

Sketch refinement is done transparently by the computer. The engineer superimposes structural grids at precise locations on top of the sketches and locates few supporting walls. Then, StAr verifies the correspondence between precise supports and imprecise architectural elements.

Structural alternatives are produced efficiently from sketches and feedback to the architect provided during design exploration. In the example, a second structural solution is provided to the architect that can motivate further design explorations.

Nevertheless, the following limitations need to be addressed in the future:

- The example building is simple in shape and size. Therefore, no problems in the sketch interpretation and refinement were found since all but two architectural elements fell within the structural grids. The interpretation algorithms need to be tested with larger and more complex buildings.

Larger buildings will enable testing the scale versus tolerance factors and more complex buildings the robustness of the recognition algorithms.
- The sketches used in the example are not so imprecise since the strokes are closely aligned with the grid lines. The tolerance factors and algorithms need to be tested with less "precise" sketches.
- The information model is static as it supports only one architect-engineer iteration. As a consequence, a common building model does not evolve as a result of structural feedback. Work is in progress to develop a model integrating the architecture and the structure for integrated early building design that evolves as a result of design decisions.

It is advisable that architects include structural considerations in their initial design explorations so that engineers are not forced to work around already made decisions. This cannot be achieved with existing structural engineering packages because they require a precise and complete model of the building to perform analysis. This paper proposes an integrated approach to conceptual design that gives an opportunity for engineers to participate in the design process at exploration time. The expected result of this early intervention is a more integrated and improved building performance.

9. Conclusions

This research is a first attempt to provide computer support for structural design from architectural sketches. It proposes an integrated approach to incorporate structural engineering concerns into architectural schematic designs earlier than traditionally done. The approach aims to provide the means for architects and engineers to refine and formalize architectural sketches into precise integrated building models efficiently, while working in their natural design environments. The first stage combines two software prototypes, one for schematic architectural design and the other for conceptual structural design, an information exchange model, and sketch interpretation mechanisms to refine sketches and make them apt for conceptual structural design. An example with a simple building demonstrates the advantages of the approach. Further tests are required with more complex buildings. Further work is also required to develop an integrated model that supports design iterations that enable evolving a building model during conceptual design.

Acknowledgements

The authors wish to acknowledge the "Commission Mixte Quebec/Wallonie-Bruxelles" for financing this research collaboration as well as the Canada Research Chair Program (for the Chair in Computer-Aided Engineering for Sustainable Building Design hold by Hugues Rivard).

References

Alvarado, C and Davis, R: 2001, Preserving the freedom of paper in a computer-based sketch tool, *Proc. of Human Computer Interaction International 2001*, pp. 687-691.

Autodesk: 2005, Available Online: http://usa.autodesk.com/, last accessed: November 2005.

CIMsteel: 2005, Steel Construction Institute (SCI), *Online previews of CIMsteel integration standards release 2*, Available Online: http://www.cis2.org/documents/cis2_docs.htm, last accessed: December 2005.

Goel, V: 1995, *Sketches of Thought*, MIT Press, Cambridge, MA.

Goldschmidt, G: 1991, The dialectics of sketching. *Design Studies* **4**: 123-143.

Gross, MD and Do, E: 1996, Ambiguous intentions: A paper-like interface for creative design, *Proc. ACM Conference on User Interface Software Technology (UIST) '96*, Seattle, WA. pp. 183-192

IAI: 2005, International Alliance for Interoperability, *Extension projects for IFC*, Available Online: http://www.iai-international.org/projects/extensionprojects.html, last accessed: December 2005.

IdeCAD: 2005, Integrated software for structural design, Available ONline: http://www.idecad.com/, last accessed: November 2005.

Khemlani, L: 2005, AECbytes product review: Autodesk revit structure, Available Online: http://www.aecbytes.com/review/RevitStructure.htm, last accessed November 2005.

Leclercq, P: 1999, Interpretative tool for architectural sketches, *1st International Roundtable Conference on Visual and Spatial Reasoning in Design: Computational and Cognitive Approaches,* MIT, Cambridge, USA, pp. 69-81.

Lecomte, A: 2003, *De l'esquisse d'architecture au predimensionnement structurel*, Travail de fin d'études en vue de l'obtention du grade d'ingénieur civil architecte, Faculté des sciences appliquées, Université de Liège.

Lipson, H and Shpitalni, M: 2000, Conceptual design and analysis by sketching, *AIEDAM* **14**: 391-401.

LuciD: 2005, Internet website of the Lucid Group, Available Online: http://139.165.122.58/Lucid/, last accessed: November 2005.

Meniru, K, Rivard, H, and Bédard, C: 2003, Specifications for computer-aided conceptual building design, *Design Studies* **24**(1): 51-71.

Miles, JC, Cen, M, Taylor, M, Bouchlaghem, NM, Anumba, CJ and Shang, H: 2004, Linking sketching and constraint checking in early conceptual design, *in* K Beucke (ed) *10th Int. Conf. on Computing in Civil and Building Eng*.

Mora, R, Rivard, H, Bédard, C: 2006a, A computer representation to support conceptual structural design within a building architectural context, *Journal of Computing in Civil Engineering,* ASCE, (to appear).

Mora, R, Bédard, C, Rivard, H: 2006b, Geometric modeling and reasoning for the conceptual design of building structures, *Journal of Advanced Engineering Informatics,* (submitted).

Schodek, DL: 2004, *Structures*, Prentice Hall, Upper Saddle River, New Jersey.

Sezgin, TM, Davis, R: 2004, Scale-space based feature point detection for digital ink, *Making Pen-Based Interaction Intelligent and Natural*.

Zeleznik, R, Herndon, K, Hughes, J: 1996, SKETCH: An interface for sketching 3D scenes, *Proceedings of SIGGRAPH 96 conference*, pp. 1-6.

DESIGNWORLD: A MULTIDISCIPLINARY COLLABORATIVE DESIGN ENVIRONMENT USING AGENTS IN A VIRTUAL WORLD

MICHAEL ROSENMAN, KATHRYN MERRICK, MARY LOU MAHER
University of Sydney, Australia

and

DAVID MARCHANT
Woods Bagot, Sydney, Australia

Abstract. This paper presents a 3D virtual world environment augmented with software agents that provides real-time multi-user collaboration for designers in different locations. This virtual world collaborative environment allows designers in different disciplines to model their view of a building as different representations. A proprietary virtual world platform augmented with software agents extends the environment to facilitate the management of the different disciplines' design representation. Agents monitor the virtual world and create, manage and display the different views of a design, and create and manage the relationships between these views. A synchronous design session with an architect and engineer demonstrates the capability and potential for the augmented virtual world for conceptual design.

1. Introduction

Large design projects, such as those in the AEC domain, involve collaboration between designers from many different design disciplines in varying locations. Existing modelling tools for developing and documenting designs of buildings (and other artefacts) tend to focus on supporting a single user from a single discipline. While this allows designers to synthesise and analyse models specific to their own discipline, it is inadequate for multi-discipline collaboration.

The complexity of building design leads to two conflicting requirements: the ability of the different disciplines to work on their part of the project using their own specific models, and the ability to communicate and

J.S. Gero (ed.), Design Computing and Cognition '06, 695–710.
© 2006 *Springer. Printed in the Netherlands.*

negotiate with the other disciplines on the synthesis and integration of the different design models. Two approaches for addressing the need for a virtual environment in which designers can coordinate domain-specific and integrated models are: a multi-user CAD system and a multi-user virtual world. While the CAD system approach uses a familiar modelling environment, CAD was not designed to be a multi-user environment and the models tend to be specific to one discipline. We propose that a virtual world approach has more potential in providing a flexible approach for modelling and communication that is not discipline specific.

The creation of different discipline models and the creation of relationships between the objects in the different models are central to the maintenance of consistency between the models. Creating these relationships requires communication between the different disciplines that can be facilitated with shared 3D visualisation, walkthroughs and rendering of the various views of the design as modelled by the different disciplines. This is of special importance at the conceptual stage of the design since much of the early collaborative decision-making is carried out at this stage. A virtual world environment based on an underlying object-oriented representation of the design is presented here as an environment that specifically supports synchronous collaboration for multiple disciplines in the design of buildings. This is in contrast to the decision made by Lee et al. (2003) to use a commercial CAD system for visualisation. One of the advantages of virtual world environments is that they allow users to be immersed in the 3D model, allowing for real-time walkthroughs and collaboration (Savioja et al. 2002; Conti et al. 2003). Moreover, CAD models contain a great deal of detail which makes real-time interaction extremely difficult.

This paper introduces DesignWorld, a prototype system for enabling collaboration between designers from different disciplines who may be in different physical locations. DesignWorld, shown in Figure 1, consists of a 3D virtual world augmented with web-based communication tools and agents for managing the different discipline objects. Unlike previous approaches which use a single shared data model (Wong and Sriram 1993; Krishnamurthy and Law 1997), DesignWorld, uses agent technology to maintain different views of a single design in order to support multidisciplinary collaboration. This architecture enables DesignWorld to address the issues of multiple representations of objects, versioning, ownership and relationships between objects from different disciplines. DesignWorld supports conceptual design where concepts are general and still fluid. Figure 1 shows two avatars representing designers who are aware of each other as well as of the various 3D models that have been constructed. The designers can build their design collaboratively, using 3D objects in real-time and explore the consequences of these designs in various 3D views.

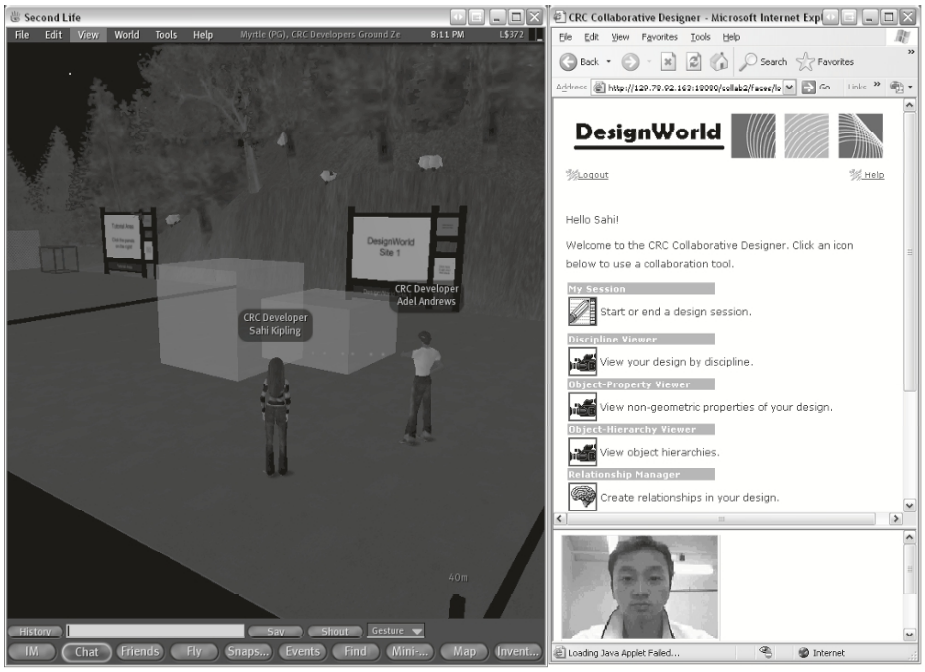


Figure 1. DesignWorld consists of a 3D virtual environment (left) augmented with web-based communication and design tools (right).

2. 3D Virtual Worlds

A virtual world is a distributed, virtual space where people can interact with other people, objects or computer controlled agents using an avatar. Moreover, the worlds are based on object-oriented modelling concepts that concur with developments in CAD and 3D modelling software. As such, they provide a suitable platform for design and collaboration. DesignWorld uses the Second Life (www.secondlife.com) virtual environment as the platform for design and collaboration. However, while virtual worlds such as Active Worlds (www.activeworlds.com) and Second Life (www.secondlife.com) offer tools for creating and modifying virtual buildings and other artefacts, they do not offer features for managing multiple representations, versions or relationships necessary for multidisciplinary design. DesignWorld addresses this issue by augmenting Second Life with web-based tools and using agents to create views and relationships and manage versions on behalf of designers. DesignWorld is an improved version of the CRC Collaborative Designer (CCD) prototype (Rosenman et al. 2005). CCD was implemented using the Active Worlds

virtual world platform. This new version, implemented in Second Life, provides facilities for modelling objects in the world and additional programming capability for associating objects in the world with an external data model.

3. Multidisciplinary Modelling

Different disciplines have different views of a design object (building) according to their functional concerns and hence create different representations or models of that object to suit their purpose. For example, a building may be viewed as: a set of activities that take place in it; a set of spaces; a sculptural form; an environment modifier or shelter provider; a set of force resisting elements; or as a configuration of physical elements. Depending on the view taken, certain objects and their properties become relevant. For the architects, floors, walls, doors and windows, are associated with spatial and environmental functions, whereas structural engineers see the walls and floors as elements capable of bearing loads and resisting forces and moments. Hence, each will create a different model incorporating the objects and properties relevant to them. Both models must coexist since the two designers will have different uses for their models. According to Bucciarelli (2003) "There is one object of design, but different object worlds." and "No participant has a 'god's eye view' of the design."

A single model approach to representing a design object is insufficient for modelling the different views of the different disciplines (Rosenman and Gero 1996, 1998). Each viewer may represent an object with different elements and different composition hierarchies. While architects may model walls on different floors as separate elements, the structural engineers may model only a single shear wall encompassing the three architect's walls. Each discipline model must, however, be consistent vis-a-vis the objects described. While Nederveen (1993), Pierra (1993), Sardet et al. (1998) and Naja (1999) use the concept of common models to communicate between the discipline models, it is never quite clear who creates the common models and maintains the consistency between them and the discipline models. In this project, this consistency will be provided by interrelationships between the various objects in different disciplines modelled by explicit (bidirectional) links from one object to another. Figure 2 shows an example of this approach, with each discipline labelling its objects according to its need and corresponding objects associated with '*correspondsTo*' relationships. While this approach may have the disadvantage of replicating information about the same object in two object models, it saves the complexities of creating the common concepts and allows each discipline great flexibility in creating its model. The discipline models allow each

discipline to work according to its own concepts and representations. The whole model may be seen as the union of the different models.

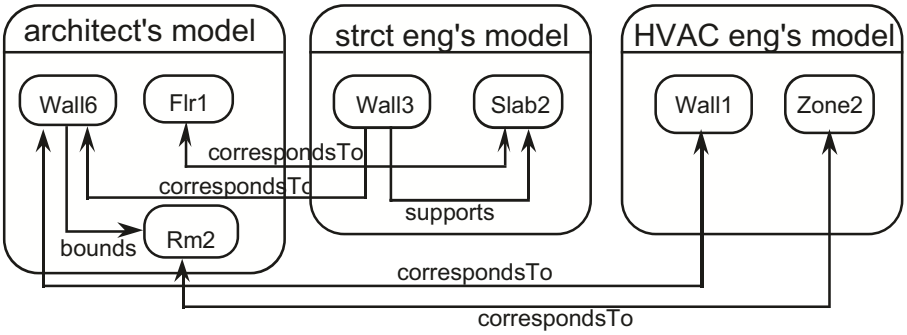


Figure 2. Discipline models and relationships.

4. DesignWorld

DesignWorld consists of three main components, the client browsers, the web applications and the external model, Figure 3.

4.1. CLIENT BROWSERS

There are two client browsers, the Second Life browser and the Web browser which provide the extended capabilities to the Second Life virtual environment. Second Life provides the environment where the different designers meet as avatars and construct their design models. The Web browser provides access to the relationships browser and the extended communications facilities. The relationships browser allows for the creation of relationships between the different objects by any of the designers. The non-geometric property browser allows for the display of information about the design objects such as the discipline that the object belongs to and the relationships with other design objects.

4.2. WEB APPLICATIONS

The web applications include the agent society, the webcam and audio facility which allow visual and aural communication, and the GroupBoard sketch tool (www.groupboard.com).

4.2.1. The Agent Society

The term *agents* in this paper refers to software agents. We take the approach that agents are systems which can sense their environment using sensors, reason about their sensory input and affect their environment using effectors. In addition, agents are systems which perform tasks on behalf of

others. Agent sensors monitor the Second Life, SQL and web environments for changes or requests from human designers. Environmental changes are stored as sensations in iconic memory, a brief store which holds sensations until they are replaced with new sensations (Ware 2000). In addition to the 'button-click sensor' which senses requests from designers through the web interface, DesignWorld agents have 'add-object sensors' to sense when objects are added to the 3D world, 'delete-object sensors' to sense when objects are deleted, 'changed-object sensors', 'selected-object sensors' and 'SQL-query sensor' to sense the external model.

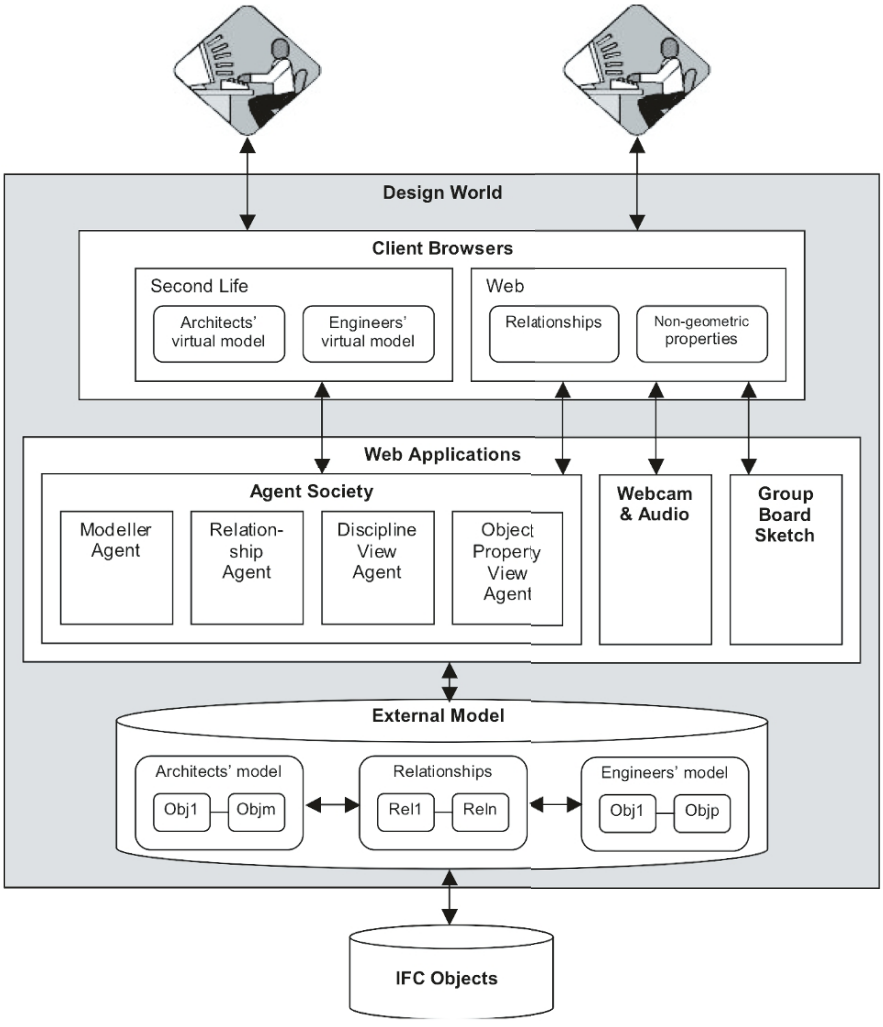


Figure 3. DesignWorld Architecture.

When requests from designers are sensed, they are combined with the most recent sensations from other sensors and sent to the agent's reasoning process. DesignWorld agents use a reflexive reasoning process which uses predefined rules to select an action to perform based on the contents of the most recent sensor data as shown in Figure 4. Where Maher and Gero (2002) propose an agent-based virtual world in which each object is an agent capable of reasoning about its environment, our agents are associated with the different tasks needed to support collaborative design in a virtual world. Maher and Gero propose three levels of reasoning for agents in virtual worlds: reflexive, reactive, and reflective. In our agent-based virtual world, we only consider the reflexive level of reasoning, although other levels of reasoning may be considered in future.

Once an action has been chosen, it triggers effectors to modify either the Second Life or web environments. Effectors include, 'change-object effectors', 'show-dialog effectors', 'SQL-database-update effectors' and 'update-web-page effectors'. These effectors allow DesignWorld agents to modify the Second Life, SQL and web environments on behalf of designers.

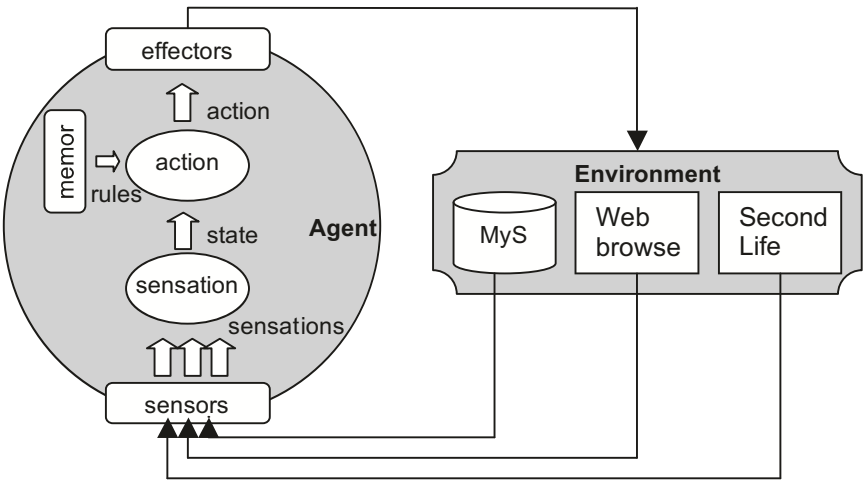


Figure 4. The DesignWorld Agent Model.

The agents in DesignWorld extend the platform of a virtual world by maintaining a model of designed objects in an SQL database in addition to the model maintained by the virtual world server. The use of an external model makes it possible to store information about design projects other than the spatial and rendering properties of individual objects stored on the virtual world server. The DesignWorld external model contains project information for a group of objects, and for each object there is discipline, versioning and

relationship information. The external model is compatible with Industry Foundation Classes (IFCs) (IAI 2000) providing the potential for models to be uploaded from IFC compatible applications such as ArchiCAD for use in collaborative sessions.

The agents in DesignWorld keep track of the objects created by each discipline in order to maintain information relevant to the different functional concerns of designers from different disciplines. A selection of viewing tools enables designers to view the components relevant to them. The agent society is comprised of the four agents, the modeller agent, the relationships agent, the discipline view agent and the object property view agent.

4.2.1.1. Modeller Agent

The Modeller agent facilitates the presentation of different views of a design by constructing and maintaining a data model of the design artefacts in a SQL database (the external model). This persistent model is capable of describing more properties of an object than can be represented in the 3D environment. For example, in Second Life an object may have an owner but the SQL external model might additionally specify a project and a design discipline to which the owner and the object belong.

The modeller agent acts upon receiving a message from the Web browser containing a request from a user for a particular design to be modelled. It retrieves information from the external model to associate non-geometric information with every object in the current Second Life environment. The modeller agent then affects the external model by writing records describing the state of the model in Second Life. The modeller agent is also responsible for maintaining different versions of a design. Each time the modeller agent affects the external model it stores records as a new version.

4.2.1.2. Relationships Agent

The relationships agent allows the designers to create and view the associations between different objects. Currently, the relationships which are supported are: *correspondsTo*, *decomposes, supports, adjacentTo* and *bounds*. The correspondsTo relationship allows the association of objects in different discipline models so as to say that they are the same object but may have different non-geometric and non-physical properties. For example a wall in the architect's model may be the same as a wall in the structural engineer's model. The wall has the same shape, dimensions and materials but its function for the architect may be to provide privacy to a space whereas its function for the structural engineer may be to support a slab. The decomposes relationship provides an association between a complex object and its components. This may also exist between objects in different

disciplines. For example, a single wall object in the structural engineer's model may be associated with three walls (one above each other) in the architect's model. The bounds relationship provides for bounding associations between objects. For example, in the early conceptual design stages, an architect may only create spatial objects, whereas a structural engineer may create wall and slab objects. The relationship between the structural engineer's objects and the architect's object will be through a bounds relationship, e.g. Wall1 (engineer object) bounds Space1 (architect object).

A relationship is created by selecting a relationship type and then selecting two objects in the relevant models. Figure 5 shows the DesignWorld interface for creating relationships. On the left is the second Life window showing a wall in the engineer's model. On the right is the Web browser window showing the creation of a bounds relationship between that wall and a space object in the architect's model.

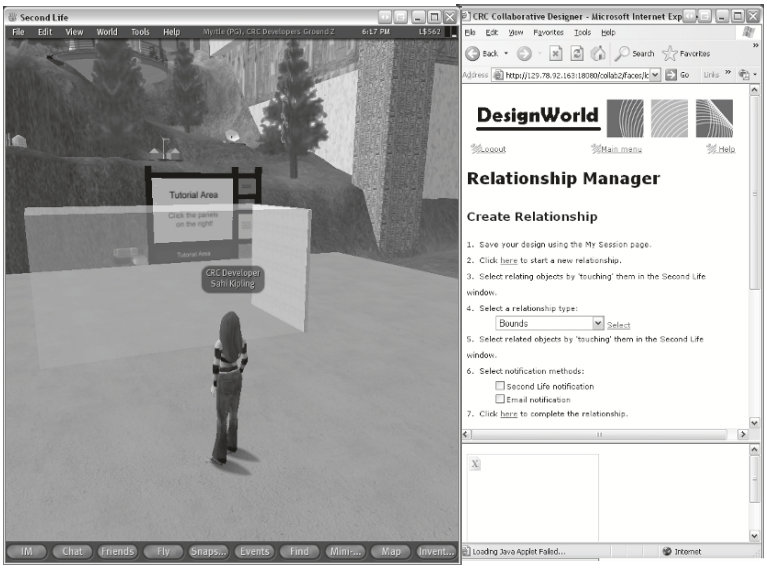


Figure 5. The Relationships Manager.

4.2.1.3. Discipline View Agent

The discipline view agent creates and displays the views of an object in Second Life as relevant to a particular discipline. A user may request a particular view in the web browser and the agent builds the view according to the objects belonging to that discipline. The Discipline Viewer Agent presents different views of a design relevant to designers from different

disciplines by retrieving relevant information from the SQL external model and modifying the design displayed in the 3D virtual environment window.

4.2.1.4. Object Property View Agent

This agent allows designers to view those non-geometric properties of objects which are not visible in the Second Life interface. These properties, stored in the external model, are displayed in the Web browser. At present, the non-geometric properties that can be attached are the discipline to which the object belongs and the relationships associated with that object, Figure 6. These properties are attached by Design World. At present, properties are not imported from the IFC model but will be in the future.

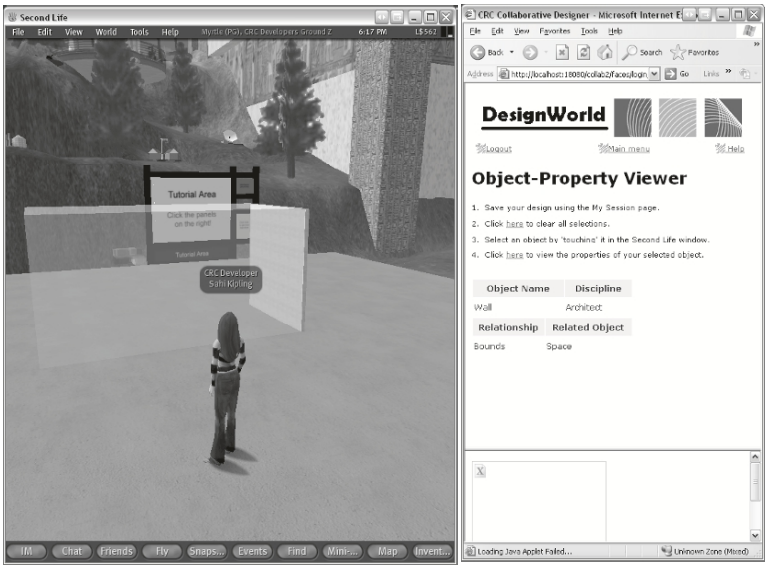


Figure 6. The object property viewer displays non-geometric properties of objects.

4.2.2. Communication Tools

Typically, avatars communicate in 3D virtual worlds using chat. This becomes inadequate in designs situations where there is a need to convey complex ideas while manipulating objects in the design. DesignWorld offers video and audio transmission facilities to support communication during design.

4.2.3. Sketching

While designers can collaborate on the 3D model of the design in the virtual world, many design ideas cannot be expressed in a 3D model. DesignWorld provides a sketching tool that allows designers to share their design ideas before committing them to a change in the 3D model. This part of the

environment uses the GroupBoard (www.groupboard.com) sketching tool. This tool enables designers to draw on a blank page, or over a snapshot of the site or current 3D model.

4.3. THE EXTERNAL MODEL

The external model is a SQL database which stores the information regarding the models and relationships. At present it allows for the extension of the geometric properties of objects created in Second Life to accommodate non-geometric properties. In the future, the external model will provide a filter to DesignWorld from the IFC model created from CAD systems. It will simplify the information in the IFC model so as to be more useful to DesignWorld. Additionally, it will allow the transfer of information derived from the creation or modification of objects in DesignWorld to be stored and transferred to the IFC model and hence back to the various designers' CAD models.

5. Collaborative Designing In DesignWorld

A designer is assigned a membership in a discipline group, e.g. architect, structural engineer, etc. Any objects constructed by that designer are assigned to that discipline group. Any designer can view any model through the view facility or a combination of view by making models transparent or not. However, designers can only modify objects that they own or which other designers have permitted them to modify. A designer can invoke the relationship facility and create, modify or delete relationships by selecting the type of relationship and the objects related. These objects may be in the same discipline model or in a different discipline model. When designers want to make a modification to an object, they will be notified of any existing relations to other objects by a dialog box as shown in Figure 7. They can then discuss the ramifications of such modifications with the appropriate discipline designer.

5.1. A COLLABORATIVE SESSION

A collaborative design task was given to two designers: an architect and a structural engineer to design an observation tower. The architect is a professional architect and the engineer is an academic. A design brief was given as follows:

> to design an innovative tower/viewing platform to fit in with an existing CRC Resource Centre designed loosely in the style of the "Mondrian School". The tower must be high enough to provide a view over the water. There are no floor space ratio or height restrictions applicable for this project.

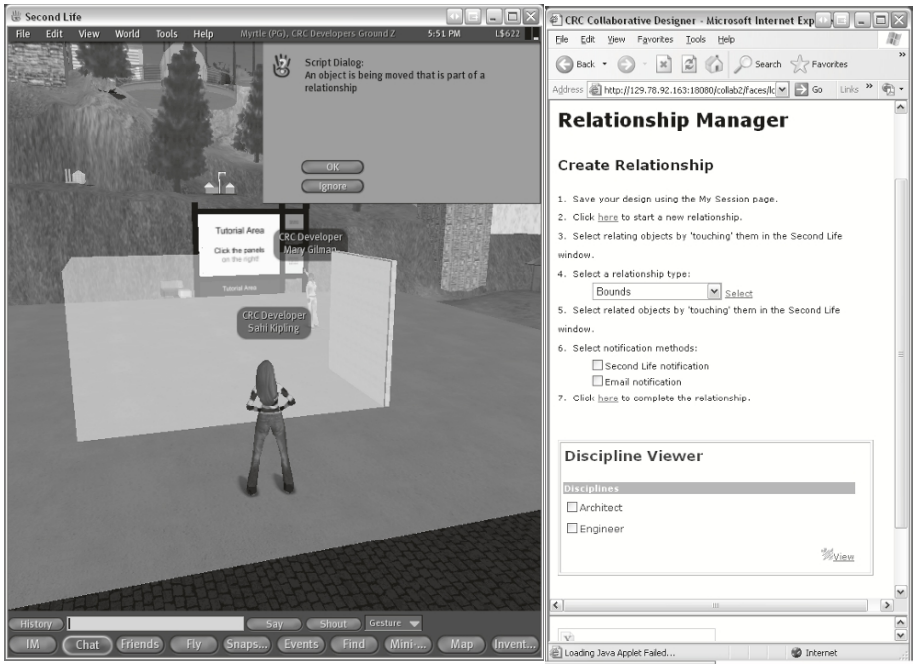
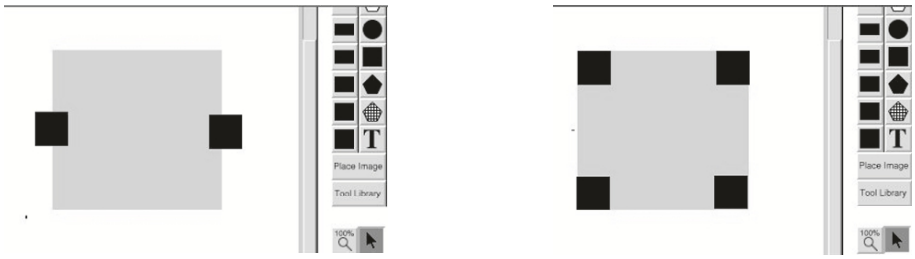


Figure 7. Designers are notified by a dialog box when they modify an object that it is part of a relationship.

In the first session, the architect and the engineer worked together in DesignWorld for 1 hour and 10 minutes. Then, 4 days later, the architect developed a more detailed design in 1.5 hours by himself, and finally, they collaborated with each other to finish their design in 2 hours at the last session of experiment. The reason for stopping the experiment in the first and second sessions is because the designers were out of time. In order to finish the design task, they spent 2 hours discussing and modelling in the DesignWorld environment. We observed the designers' sessions and collected video and voice from the sessions. In this paper we report only on our observations of the design sessions and the designers' perception of their productivity. Additional analysis of the data collected from the session is not within the scope of this paper.

The first session began with the architect and the engineer discussing the brief and using the GroupBoard to set out the first concepts. The architect suggested using two towers opposite each other to support a 10 m x 10 m platform, Figure 8a. The engineer pointed out that while the structure would be stable in one direction it would not be stable in the other and that either the towers would have to be increased in size in the less stable direction or that each tower could be split in two and separated to provide sufficient

stability. The architect, at first thought about using the corners of the platform, which would provide excellent stability, Figure 8b, but decided against that, as it would impede the view to all sides.



(a) First concept (b) Second concept

Figure 8. Groupboard images of initial design concepts.

A decision was made to have two pairs of towers, each pair on either side of the platform and 2 m apart. The engineer noted that the towers would have to be connected at intermediate levels. A decision was made to provide struts between the towers in both directions. Both the towers and the struts were designed to match to the black Mondrian framing pattern. Figure 9 shows the final concept agreed on. It shows two main beams (displayed in red on the screen) projecting from the towers to support the platform and a series of mullions that would support the upper part of the platform as well as adding to the "Mondrian look". On the side is a sketch of the elevation showing two platforms, one at a lower level, and a series of struts connecting the towers.

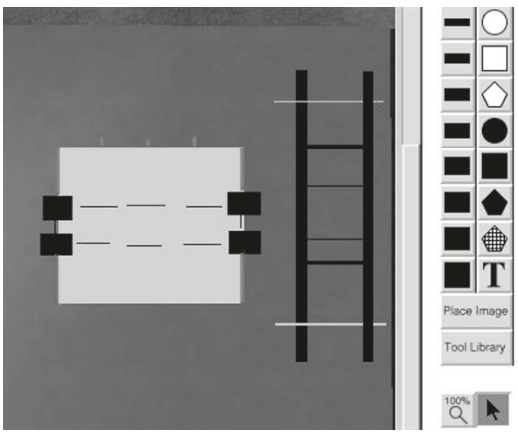


Figure 9. Final concept.

The architect then began building in 3D in the SecondLife window. In the first session, which lasted 1 hour, the four towers and the viewing platform

were constructed. The architect continued to complete his conceptual design on another day by himself by adding the upper platform space and the mullions.

In the second synchronous session, the engineer created his model by taking those objects from the architect's model which were also structural elements, i.e. the towers and the struts. He then added beams to support the floor and roof of the upper platform. The architect at this stage thought that to emphasize the "Mondrian look" presented by this framing and also to emphasize the airborne feeling of the observation space, he would make both the floor and roof transparent. This being a conceptual design, issues such as thermal comfort were left unconsidered. The structural implications of this were also left unconsidered at this stage. Figure 10 shows the design from this stage. On the left is the SecondLife architect's view and on the right is the design imported into the GroupBoard. The circled area (this was displayed in red on the designers' screens) at the bottom denotes an area circled by the engineer for discussion.

The designers completed a questionnaire at the end of each session, asking questions about productivity and their perception of which tools they used and which tools were most productive. During the synchronous session only the sketching and 3D modeling tools were used. In the synchronous sessions the architect and engineer were in voice contact but found there was no need for any video contact as no extra material was required to be shown. The designers did not use the relationship manager during the synchronous session because it took time away from further development of the design. In terms of productivity, the designers indicated that they were very productive in the first (synchronous) session. But in the second session where only the architect worked, the productivity was only moderate. In the final session, they considered their productivity to be high as far as their ability to arrive at and represent design decisions in the 3-D environment.

6. Summary

This paper presents DesignWorld, a prototype system for enabling multidisciplinary, distributed design collaboration. DesignWorld consists of a 3D virtual world augmented with a number of web-based tools for the creation of different discipline views as well as the necessary relationships between these views to provide and maintain consistency. Unlike previous approaches, DesignWorld uses agent technology to maintain different views of a single multidisciplinary project. It addresses the issues of multiple representations of objects, versioning, ownership and relationships between objects from different disciplines.

The collaborative session demonstrates the effectiveness of the DesignWorld environment in being able to quickly develop a concept and

then embody that concept in a 3D model. Both architect and engineer views exist and can be viewed and worked on separately as required. The relationships between objects provide a notification that any changes have ramifications on other disciplines.

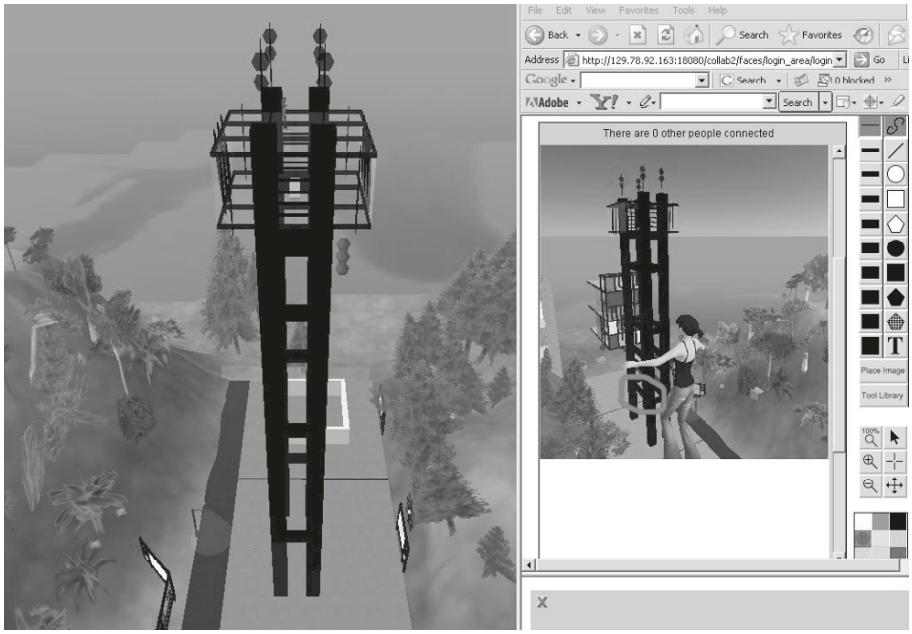


Figure 10. The tower design. SecondLife and GroupBoard views.

At present the modelling of views is implemented fully in DesignWorld. Future work will extend the capabilities of DesignWorld to receive information from, and place information in, IFC models generated from a discipline's CAD modelling. The information in the IFC model will be translated to the external model and any new information produced (or modified) in DesignWorld and stored in the external model will eventually be translated into the IFC model.

The capabilities of virtual worlds as modelling tools is still at an early stage and some improvements are necessary for these to be a simple and flexible tool for designers. For example, the GroupBoard could be replaced by a more flexible sketching tool and the Virtual World modeling capabilities will inevitably improve. Moreover, it is necessary for the users to become more familiar with the tools to exploit their capabilities. However, the above has shown that, in general, the designers were able to communicate multi-disciplinary concerns and issues and achieve design models that satisfy the requirements of both disciplines. We anticipate that the translation to and from CAD systems via IFCs (or some other standard)

will make the approach proposed by DesignWorld more useful for other stages of the design process.

Acknowledgements

The research described was supported by the Australian Cooperative Research Centre for Construction Innovation. This work is part of the Team Collaboration in High Bandwidth Virtual Environment project.

References

Bucciarelli, LL: 2003, Designing and learning: A disjunction in contexts, *Design Studies* **24**(3): 295-311.

Conti, G, Ucelli, G and De Amicis, R: 2003, JCAD-VR – A multi-user virtual reality design system for conceptual design, *TOPICS, Reports of the INI-GraphicsNet* **15**: 7-9.

IAI (2000). Industry Foundation Classes-Release 2x, IFC Technical Guide, Available Online: http://www.iai-international.org/iai_international/Technical_Documents/documentation/IFC_2x_Technical_Guide.pdf

Lee, K, Chin, S and Kim, J: 2003, A core system for design information management using industry foundation classes, *Computer-Aided Civil and Infrastructure Engineering* **18**: 286-298.

Maher, ML and Gero, JS: 2002, Agent models of 3D virtual worlds, *ACADIA 2002: Thresholds*, California State Polytechnic University, Pamona, pp. 127-138.

Naja, H: 1999; Multiview databases for building modelling, *Automation in Construction* **8**: 567-579.

Nederveen, SV: 1993, View integration in building design *in* KS Mathur, MP Betts and KW Tham (eds), *Management of Information Technology for Construction*, Singapore:World Scientific, pp. 209-221.

Pierra, G: 1993, A multiple perspective object oriented model for engineering design, in X Zhang (ed), New *Advances in Computer Aided Design and Computer Graphics*, International Academic Publishers, Beijing, pp. 368-373.

Rosenman, MA and Gero, JS: 1996, Modelling multiple views of design objects in a collaborative CAD environment, *CAD Special Issue on AI in Design* **28**(3): 207-216.

Rosenman, MA. and Gero, JS: 1998, CAD modelling in multidisciplinary design domains, *in* I Smith (ed), *Artificial Intelligence in Structural Engineering*, Springer, Berlin, pp. 335-347.

Rosenman, MA, Smith, G, Ding, L, Marchant, D and Maher, ML: 2005, Multidisciplinary design in virtual worlds, *in* B Martens and A Brown (eds), *Computer Aided Architectural Design Futures 2005*, Springer, Dordrecht, Netherlands, pp. 433-442.

Sardet, E, Pierra, G, Poiter, JC, Battier, G, Derouet. JC, Willmann, N and Mahir, A: 1998, Exchange of component data: The PLIB (ISO 13584) model, standard and tools, *Proceedings of the CALS EUROPE'98 Conference*, pp. 160-176.

Savioja, L, Mantere, M, Olli, I, Ayravainen, S, Grohn, M and Iso-Aho, J: 2003, Utilizing virtual environments in construction projects, *ITCon* **8**: 85-99, Available Online: http://www.itcon.org/cgi-bin/papers/Show?2003_7

Ware, C: 2000, *Information Visualisation: Perception for Design*, Morgan Kaufmann, San Mateo.

Wong, A and Sriram, D: 1993, SHARED An information model for cooperative product development, *Research in Engineering Design* **5**: 21-39.

CONTACT AUTHORS' EMAIL ADDRESSES

Ang, MC	menmca@leeds.ac.uk
Bandini, S	bandini@disco.unimib.it
Burge, J	burgeje@muohio.edu
Deak, P	pdeak@computing.dundee.ac.uk
Dong, A	adong@arch.usyd.edu.au
Duarte, J	jduarte@civil.ist.utl.pt
Gero, JS	john@arch.usyd.edu.au
Hanna, S	S.Hanna@cs.ucl.ac.uk
Haymaker, J	haymaker@stanford.edu
Holden, T	holden@eng.cam.ac.uk
Janssen, P	patrick@janssen.com
Kan, J	waitakkan@cuhk.edu.hk
Keller, R	rk313@cam.ac.uk
Koutamanis, A	a.koutamanis@bk.tudelft.nl
Kvan, T	tkvan@arch.usyd.edu.au
Liikkanen, L	liikkane@cc.helsinki.fi
Maher, ML	mary@arch.usyd.edu.au
Matthews, P	p.c.matthews@durham.ac.uk
Milettie, G	gpm@wpi.edu
Mora, R	rodrigo_mora@sympatico.ca
Nagai, Y	ynagai@jaist.ac.jp
Oh, Y	yeonjoo@cmu.edu
Prats, M	m.prats@open.ac.uk
Rodgers, P	p.rodgers@napier.ac.uk
Rosenman, M	mike@arch.usyd.edu.au
Rudolph, S	rudolph@isd.uni-stuttgart.de
Saariluoma, P	psa@it.jyu.fi
Schwede, D	dirk.schwede@student.usyd.edu.au
Shelton, K	ashva@comcast.net
Tenetti, R	raji@cad.strath.ac.uk
Treur, J	treur@few.vu.nl
Winkelmann, C	winkelmann@psychologie.tu-dresden.de
Yan, W	wyan@archmail.tamu.edu
Yaner, P	yaner@cc.gatech.edu

AUTHOR INDEX

Ang, MC 521
Arciszewski,T 461
Bandini, S 141
Bilda, Z 265, 305
Brown, D 183
Burge, J 655
Caldas, L 483
Chau, HH 521
Clarkson, PJ 41
Cornford, S 655
Cross, V 655
de Pennington, A 521
Deak, P 503
Do, E Y-L 123
Dong, A 385
Duarte, J 483
Ducla-Soares, G 483
Duffy, A 285
Earl, CF 83
Eckert, C 41
Gao, S 245
Gero, JS 265, 407
Goel, A 423
Gross, M 123
Gül, F 305
Hacker, W 603
Hanna, S 3, 563
Haroun Mahdavi, S 563
Haymaker, J 635
Holden, T 163
Janssen, P 365
Johnson, G 123
Jonker, C 203
Juchmes, R 675
Kalay, Y 61
Kan, J 265
Kannengiesser, U 407
Karvinen, M 325
Keller, R 41
Kerley, W 163
Kiper, J 655
Koutamanis, A 103, 345
Kvan, T 245
Leclercq, P 675
Liikkanen, L 619
McKay, A 521
Maher, ML 305, 695
Marchant, D 695
Matthews, P 223
Maynard-Zhang, P 655
Merrick, K 695
Milette, G 183
Mora, R 675
Nagai, Y 443
Nevala, K 325
Oh, Y 123
Perttula, M 619
Prats, M 83
Reed, C 503
Rivard, H 675
Rocha, J 483
Rodgers, P 583
Rosenman, M 695
Rowe, G 503
Rudolph, S 541
Saariluoma, P 325
Sartori, F 141
Schwede, D 23
Sharpanskykh, A 203
Shelton, K 461
Taura, T 443
Tenneti, R 285

Treur, J 203
Winkelmann, C 603
Yan, W 61
Yaner, P 423
Yolum, P 203