A FRAMEWORK FOR RESEARCH IN DESIGN COMPUTING

John S Gero and Mary Lou Maher

Key Centre of Design Computing

Department of Architectural and Design Science

University of Sydney NSW 2006 Australia

Ph: +61-2-9351 2328 Fax: +61-2-9351 3031

email:
john@arch.usyd.edu.au
mary@arch.usyd.edu.au

Keywords: CAAD, research, cognitive models, axiomatic models, conjectural models

1. Introduction

Design computing has often been considered a subset of computer applications that assist the designer in documenting and analysing complex designs. As one of many areas in which computer applications have beeen developed, design computing has relied on software developers and vendors to implement and market software with the relevant features and utilities to support some aspects of design activity. In this paper we consider design computing as a research area, one in which the results of the research lead to more than additional computer programs and in fact lead to a better understanding of designing and computer support for designing.

Considering design computing as a research area, we identify three sets of goals:

(i) to develop theories, models and methods of designing as a process;

(ii) to use these theories, models and methods as the basis for the development of tools;

(iii) to use these theories, models and methods as the basis for teaching.

The first set of goals has more to do with design research rather than strictly design computing research. In order to achieve the first set of goals, it is sometimes useful to consider computational models of design as a way of simulating design processes. However, human designers can also provide the basis for developing theories, models, and methods of designing. The second set of goals looks at the implications of particular theories, models, and methods of designing when considering computer support or automation of specific design tasks. This set of goals has a more direct correlation with the majority of design computing research currently taking place at universities. The third set of goals brings this understanding of design processes to bear on how we teach design. Here again, the focus is not entirely on computer applications for design, but on the use of computational models and/or

cognitive models of design to inform design teaching.

The Key Centre of Design Computing at the University of Sydney carries out teaching and research in the area of design computing. There are approximately 300 undergraduate architecture students, 60 graduate design computing students, 15-20 doctoral students, and 10 academic and research staff at the Key Centre. The framework for design computing research presented here is based on research that has taken place at the Key Centre over the last 20 years.

Design computing research can be pursued using a variety of scientific methods, the paradigms that we find to be both useful and distinctive are:

(i) empirically-based research (cognitive models);

(ii) axiom-based research (computational models); and

(iii) conjecture-based research (computational models).

(a) conjectures based on an analogy with cognitive processes; and

(b) conjectures based on an analogy with computational processes.

Empirically-based research involves the development of experimental studies of designers that result in cognitive models of designing. Axiom-based research involves the identification of a set of axioms and their consequences to derive a logic-based computational model of designing. Conjecture-based research involves an analogy between a cognitive or computational process that leads to a computational model specific to designing. This paper describes the characteristics of each of the three paradigms and gives examples of research projects at the Key Centre that illustrate the approach and preliminary results obtained through the different paradigms.

2. Empirically-Based Design Computing Research

Empirically-based research uses the experimental paradigm in which experiments are set up and then data is collected and analysed to produce a set of results. These results are then used as the basis of either the development of a hypothesis or the confirmation of a hpothesis about designing. Typically the experiments are developed to provide evidence for a particular theory or cognitive model of designing. Typical approaches to empirically-based design computing research are: direct observation of the results of designing; surveys of designers' perceptions; and protocol studies of individual and collaborating designers designing. New protocol analysis methods have been developed and are being applied to produce novel results concerning the behaviour of designers as they are designing which has significance for the development of computational tools for designers.

2.1 Protocol analysis of designers

Protocol studies are a means of obtaining data from verbal utterances. Designers are asked to "think aloud" while they are designing. While they are designing they are video- and audio-taped. The designer's verbal utterances are transcribed. The transcription is then used, along with design theory, to develop a coding scheme. The transcription in then coded and finally analysed. The steps are listed below:

- taping

- transcription
- code development
- coding
- analysis

The results of such studies provides grounded insight into the behaviour of designers as they are designing. These insights form the basis of the development of computational support tools for designers.

2.1.1 An experimental study of designers

Designers were asked to carry out a specified design task and the "talk aloud" method was employed. Each designer was videotaped and a rich coding scheme was developed based on both design theory and the need to accommodate the data in the transcription. The development of the coding scheme is a crucial aspect of the protocol analysis method. The coding scheme developed here used five generic categories. The advantage of the use of categories is that they allow for an additional confirmation phase in the analysis since they exhibit an interdependence. The five categories developed were (Gero and McNeill, 1977):

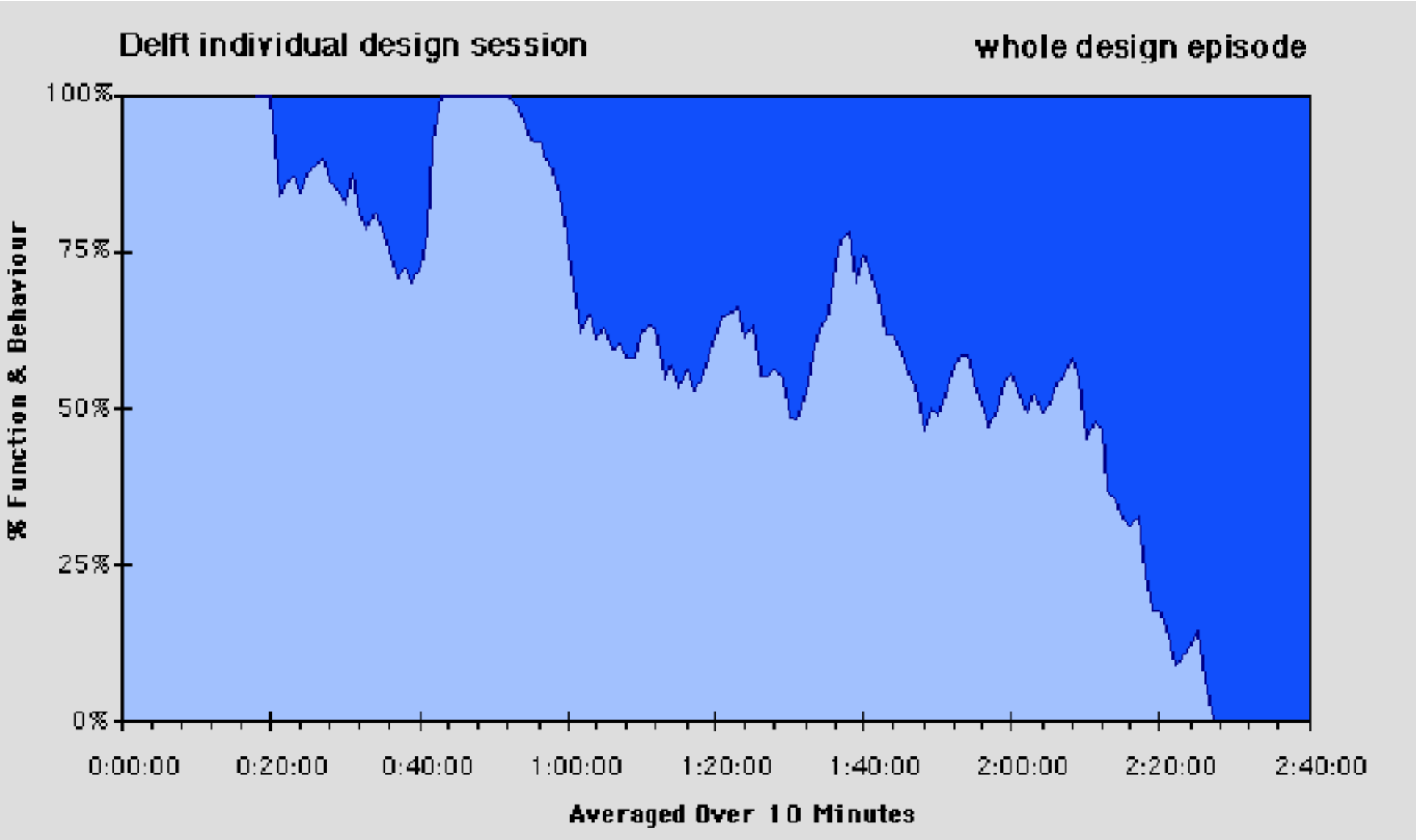
1. *problem domain - abstraction level*
    - systemic
    - system
    - component
    - component groups
1. *function-behaviour-structure*
    - requirements
    - function
    - behaviour
    - structure
1. *analysis and evaluation*
    - analysing proposed solutions
    - calculation for analysis
    - postponing analysis
    - evaluating result of analysis
    - justification
1. *synthesis micro-strategies*

- proposing solution
- clarifying a proposal
- retracting a proposal
- making a decision
- consulting external information
- postponing a design action
- looking ahead
- using application knowledge
- using domain knowledge
- referring to design strategy

1. *design macro-strategies*
    - top down
    - bottom up
    - decomposition
    - recomposition
    - backtracking
    - opportunistic

2.1.2 Protocol analysis results

At a gross level a designer's time can be spent either on postulating solutions, called structure, or in reasoning about the function and behaviour of possible or postulated designs. Figure 1 shows a typical distribution of the time spent between these two large classes of activities by a designer. It is interesting to note that it is almost twenty minutes into the session, for this design, before any structure is proposed.



The percentage of time spent in reasoning about Function and Behaviour as compared to Structure, calculated at one minute intervals and averaged over periods of ten minutes.

Figure 1. Typical plot of distribution of time spent on function and behaviour (light), as against structure (dark), for an experienced designer (Gero and McNeill, 1997).

Considerable detail about various aspects of designers' behaviour can be determined using the protocol analysis method. Figure 2 shows the spectrum of design event lengths across a typical design session. What is surprising is the very short duration of each design event. Without experiments with human designers such information would not become available.

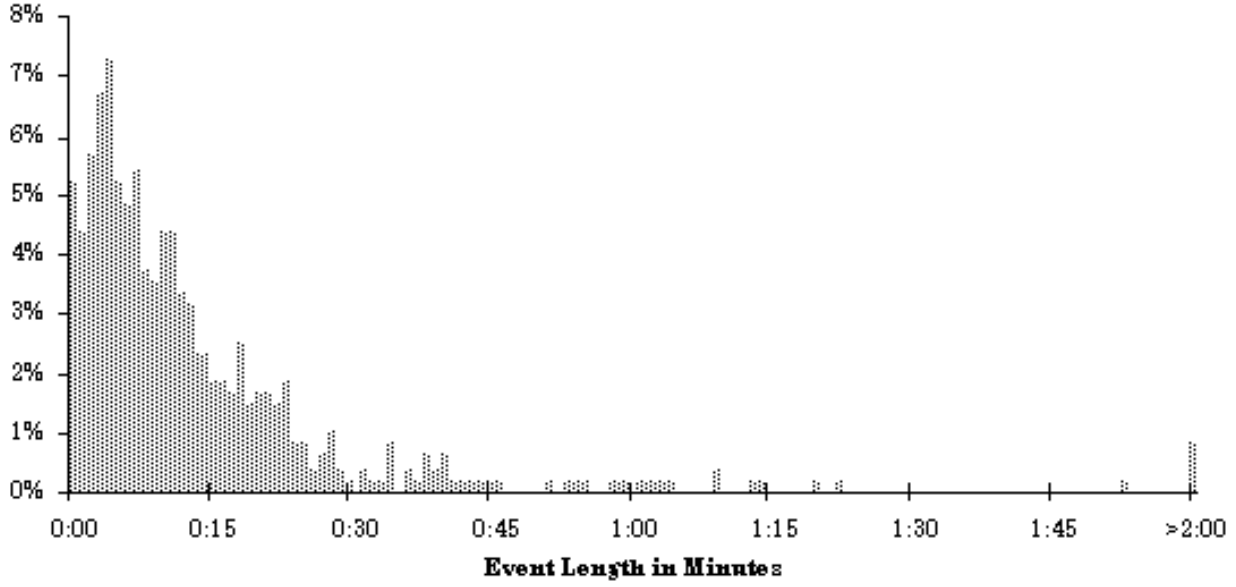


Figure 2. Spectrum of coding design event lengths (Gero and McNeill, 1997).

2.2 An experimental study of computer-mediated collaborative design

We developed an experimental study of computer-mediated collaborative design (CMCD) based on the methodology of protocol studies and analytical coding schemes to understand the difference in the documentation that results from a designer working alone as compared to designers involved in CMCD. In this study we borrow the idea of coding and analysing the data of the collaborative session from protocol analysis techniques, but we do not consider the data to be the verbal utterances of the subjects. Since we are not trying to reconstruct the collaborative design *process*, but to understand how the designers document their designs differently when collaborating, the verbal utterances would not provide the correct data. Rather, we have taken the data to be the information that has been saved as working files on the computer to document the design.

2.2.1 The CMCD experiment

The experiment includes two design sessions for each participant. In each session, designers document their designs using the computer. In the first session we established base data for each designer by asking him to design alone. In the second session we asked two designers to design collaboratively. During session 1 each designer is asked to work on Design Problem 1 (DP1) on their own for approximately two hours. During session 2, a pair of designers is asked to collaboratively solve Design Problem 2 (DP2), again for approximately two hours. DP2 is a similar type of problem to DP1, but with a different brief. In both sessions the designers use drawing packages, CAD systems, and video conference whiteboards to present their designs.

The data collected from the experiment is a derivative of the documentation produced during the two sessions. We coded and analysed the documented designs for structure and semantic content. The structure of a design is generally described by its geometry. Geometry is that part of a design in which a shape is formalised. Formal representations of geometry provide the logic and expressiveness of a mathematical language. For example, space configuration, orientation of the elements and thickness of walls are typical properties formalised in a geometric representation. However, geometry does not say anything about the purpose of the structure until the functions of design elements are described explicitly. The semantic content includes "reasons of choice". Semantics are a fundamental part of a collaborative session. Whereas image and shape can be visualised and pointed out with cursors, purpose, function and performance cannot be indicated until they find a formalisation in a drawing, i.e. until they are described explicitly.

We developed a two level coding scheme: one using a data-driven approach and the second a hypothesis- or expectation-driven approach. Using the data-driven approach, the elements of the documented designs have been counted and categorised according to their text and geometry content. Using the expectation-driven approach, we classify the categorised elements as "semantics" or "structure". We define the semantics elements as those which document the purpose of the design element and the structure elements to be those that document the geometry of the design element.

2.2.1 The CMCD results

When considering the amount of documented design semantics, we found that both the single designer and the collaborative designers recorded very little semantics. We also found that the amount of semantic information documented is less than the amount of geometry, as illustrated in Figure 3.

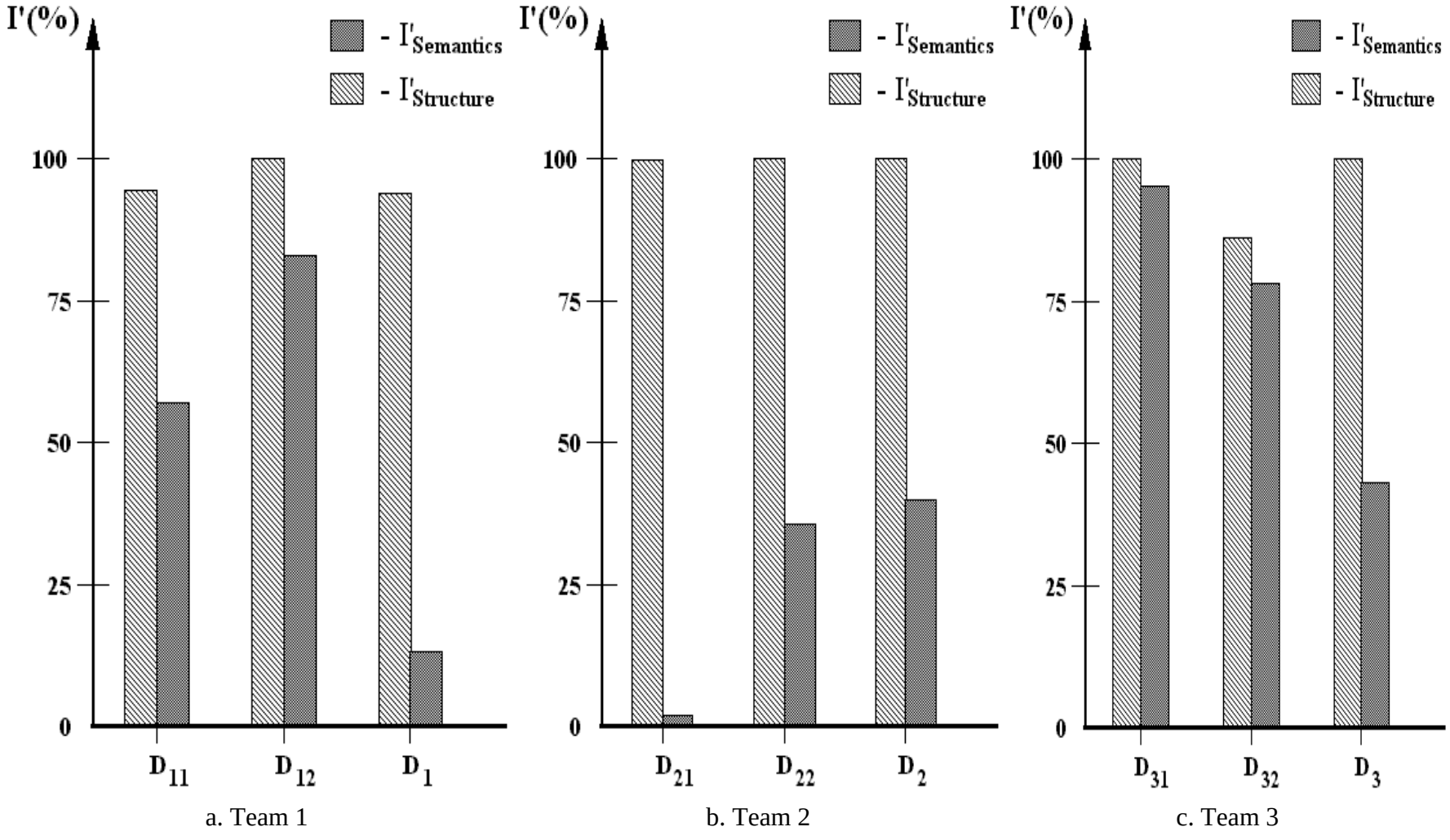


Figure 3. Comparing documented semantics and structure for all designs.

The data we collected in this experiment was the documented designs. Additional data that was not collected and analysed was the verbal utterances of the designers. A significant amount of design semantics was communicated in conversation. Based on our observations of the designers while collaborating we found that due to the intensive information exchange via video conferencing between the parties during a CMCD session, a valuable amount of the semantic information is left undocumented. Designers described their design semantics verbally, through video and audio channels, and this information is not included in the final design document.

The methodology of establishing two sessions for each designer to compare the effect of collaboration on design activity is a general methodology that can be applied to other studies. We found that by establishing base data for each designer, we could isolate the effect of collaboration on the resulting design documentation. Other applications of this methodology could be the study of the design process, the effect of negotiation, and the establishment of design styles.

3. Axiom-Based Design Computing Research

Axiom-based research produces computational models of design through the identification of a set of axioms and the logical consequences of the axioms. This approach to design computing research involves:

- specifying relevant axioms
- deriving logical consequences of the axioms
- mapping the axioms and their consequences onto a particular domain to derive new results.

For example, an axiomatic logic-based shape representation allows for the uniform representation of shapes with or without curved boundaries, the consequences of which are representations of complex shapes that can be manipulated with logical implications (Damski and Gero, 1996). Consider the universe of discourse as the space defined in Figure 4. The axiom is that the space can be divided into two complementary spaces.

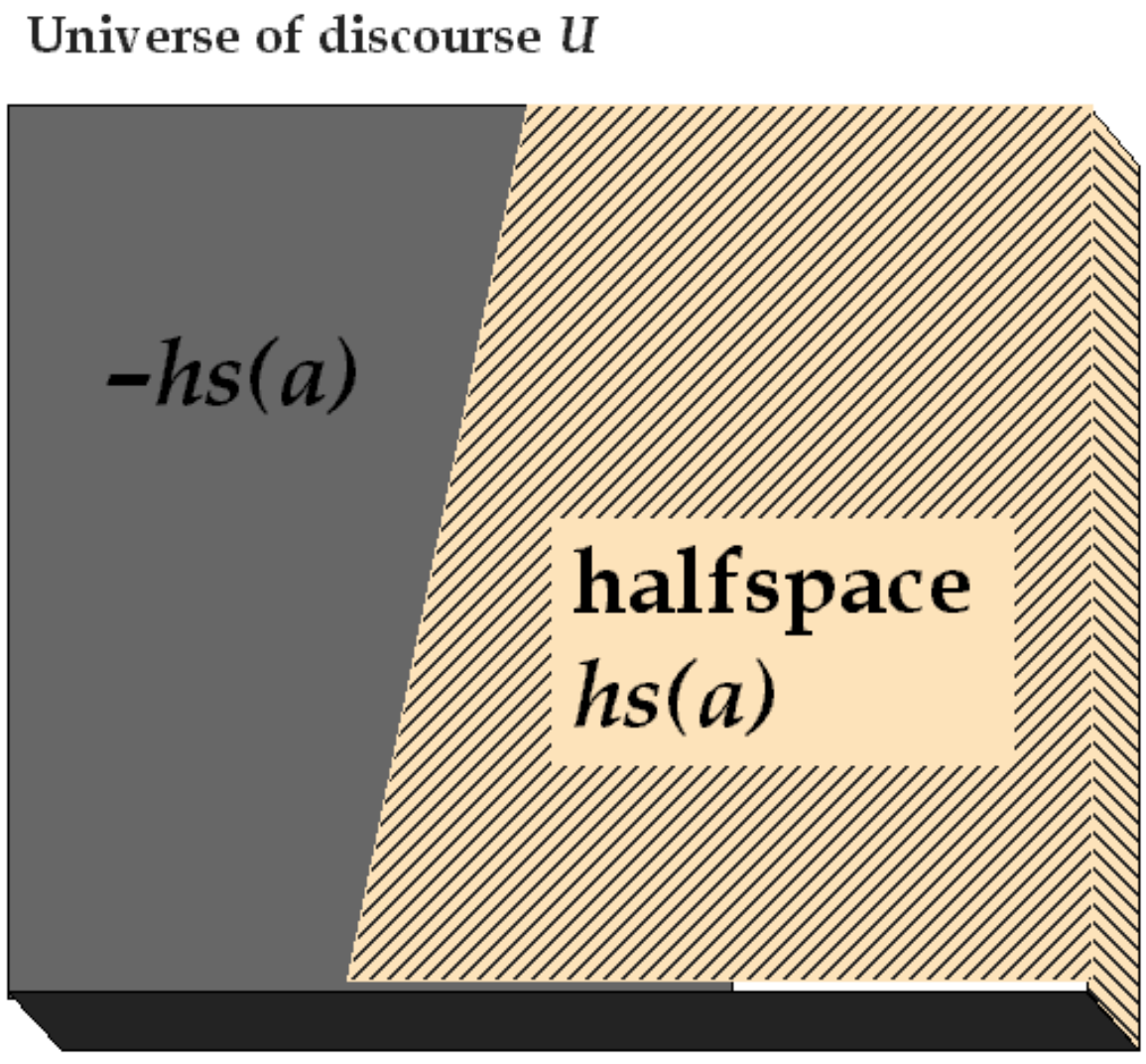


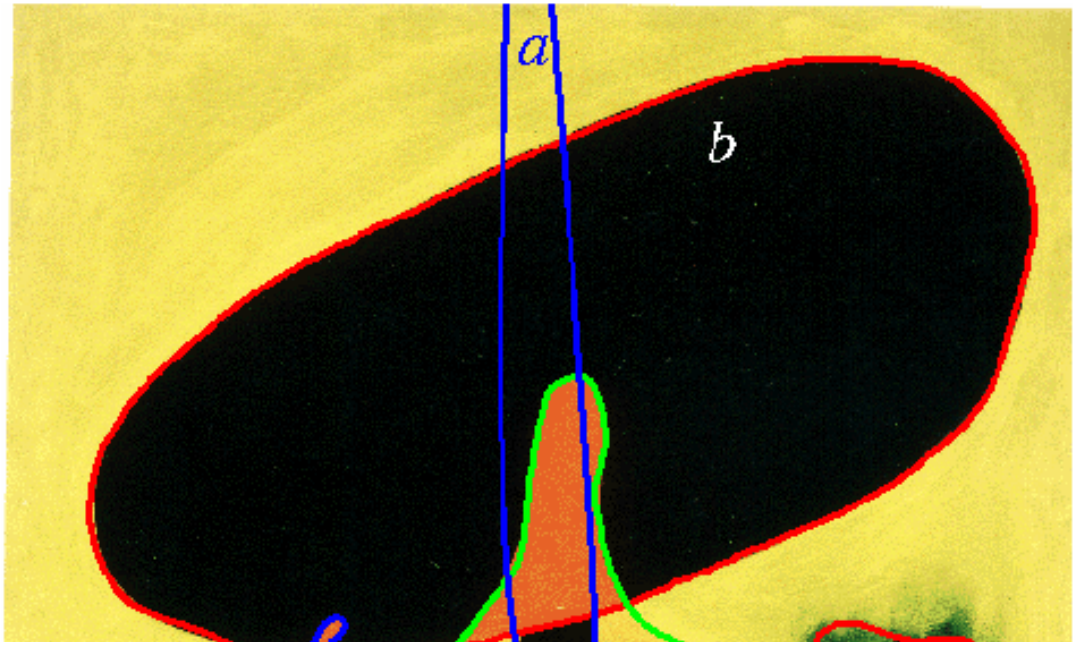
Figure 4. A space divided into two halfspaces, labelled hs(a) and -hs(a).

The following can be defined or inferred from the axiom:

- a predicate *hs(a)* is defined for the halfspace *a* and *-hs(a)* for the halfspace *a'*, *hs(a)* is defined as True and *-hs(a)* as False

- a volume *V* is *hs(a1)* Λ *hs(a2)* Λ..... Λ *hs(an)*
- a shape S is *V1* / *V2* / *V3* / ..... / *Vm.*

Consider the painting in Figure 5 which shows a girl with a hat, along with a set of labelled halfplanes. The representation of such near arbitrary shapes is computationally extremely difficult if the designer wishes to reason further about them. The axiomatic approach described here can handle these shapes.



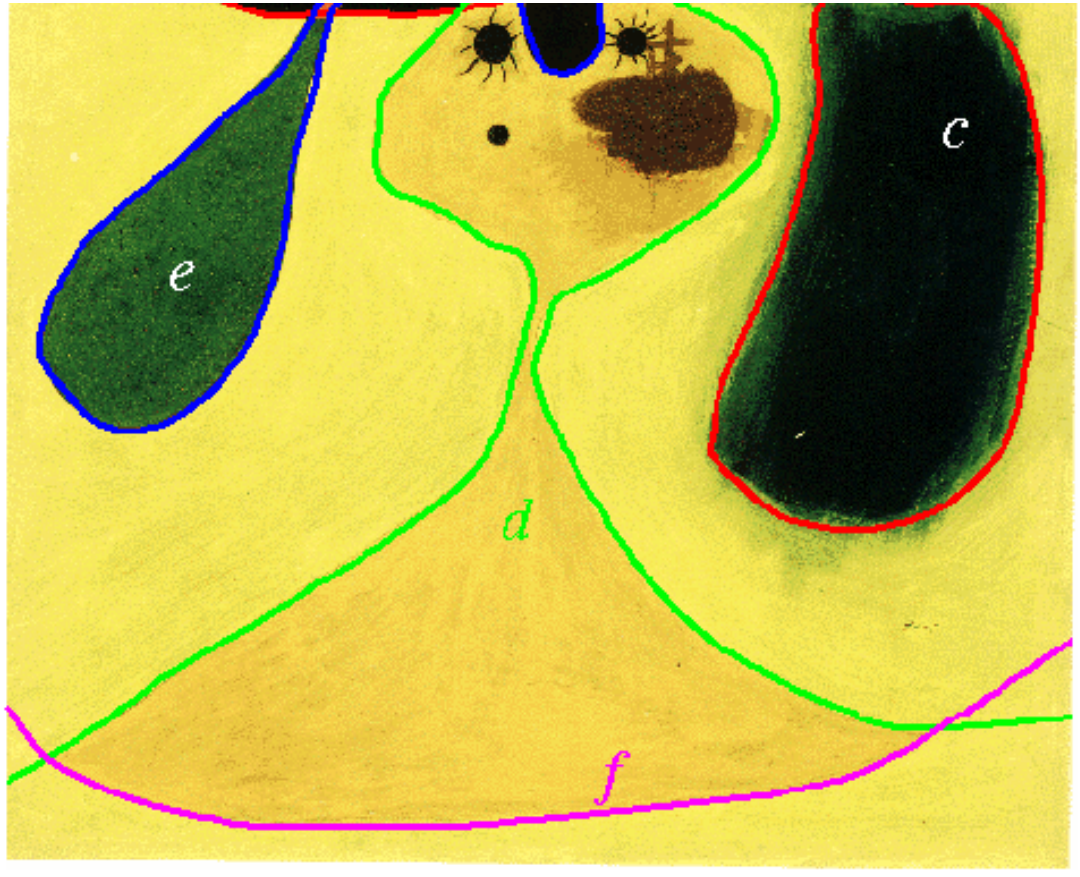


Figure 5. Miro's Girl in a Hat.

The girl's hat is defined by:

$$hp(b) \wedge \neg hp(c) \wedge hp(f)$$

The girl's head and body is defined by:

$$\neg hp(c) \wedge hp(d) \wedge \neg hp(e) \wedge hp(f)$$

From such representations we can carry out a variety of shape and topological computations even though the original shapes are difficult to represent numerically.

4. Analogy-Based Design Computing Research

The development of theories, models and methods of designing often relies on identifying an analogy with other processes. This research paradigm starts with a relevant computational process or cognitive model of design and develops a specific computational model of design. Some examples of computational models based on an analogy with cognitive models of design include: case-based design (design based on precedents; representation of cases including multimedia representations); design prototypes (knowledge chunking); graphical emergence (emergence of shapes, objects, semantics and style from drawings); design by analogy (between domain analogies in particular); and qualitative reasoning in design (qualitative representation and reasoning about shapes and spaces). The development of computational models of designing need not rely entirely on cognitive models of designers, there is the potential to identify an analogy with other computational processes and apply them to a design domain. This type of research borrows heavily from computing fields such as AI and Operations Research to produce specific computational models of design; for example: evolutionary systems (genetic engineering and co-evolution); and neural networks (emergence models).

4.1 Case-based design

Case-based reasoning provides design support by reminding designers of previous experiences that can help with new situations. As designers, we learn to design by experiencing design situations. This is reflected in the way we teach students to be designers: engineers are taught to have analytical capabilities and then learn to design in professional practice, architects are taught through exposure to a range of design experiences in the studio. We learn to analyze through the use of formal methods, but creating a new design requires previous experience, or at least, exposure to another's design experiences. As a cognitive model of design, case-based reasoning provides the basis for a computational model of design, as illustrated in Figure 6.

Case-based reasoning as a support environment for conceptual design is attractive for two reasons:

1. the knowledge is represented as design cases that can be proprietary and/or familiar to the designer, and

2. the knowledge as case memory can be maintained and updated automatically with the use of the system.

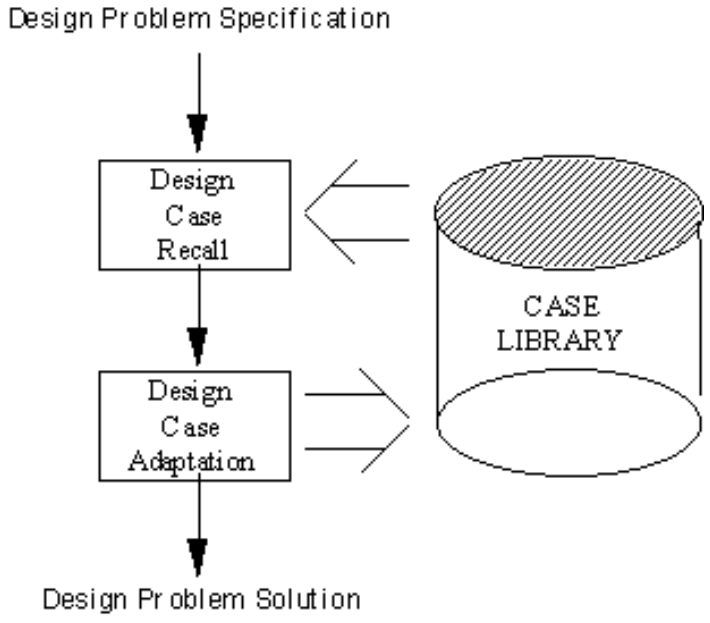


Figure 6. Computational model of design as case-based reasoning.

The application of case-based reasoning to structural design, eg CASECAD and CADSYN (Maher et al, 1995), has shown that the development of these case-based reasoning systems has to take into consideration a formal representation of design cases and the knowledge needed to adapt the design.

4.1.1 Case representation and retrieval

Design cases can be represented using the attribute-value formalism, widely used in supporting design problem solving using Artificial Intelligence techniques. New design problems are described using the same formalism: attribute-value pairs that describe requirements on the features of the new design. Additional information might also be part of a case representation, such as graphical representations, text annotations, etc. (*eg.*, see (Maher and Balachandran, 1994). This additional information helps the user's understanding of a design case, rather than being useful for automated reasoning. In order to facilitate more flexible and efficient memory retrieval (*eg.*, see (de Silva Garza and Maher, 1996)), we categorise the attributes used to describe the memory items into four classes: context, function, behaviour, and structure (a scheme adapted from Gero's design prototypes (Gero, 1990)).

Determining the relevance of a design case to the current problem-solving situation requires matching the attribute-value pairs in the problem specification with those contained in the design cases. An indexing tree organisation facilitates this matching process and the subsequent retrieval of relevant design cases from memory. Typically all design cases that share a subset of the problem specification are retrieved and then one or more of these cases is selected for adaptation.

4.1.2 Case adaptation

We have tried a number of approaches for design case adapation, including rule-based value substitution and constraint satisfaction (Maher et al, 1995). More recently we have been exploring the use of a genetic algorithm to perform design case adaptation. Genetic Algorithms (GAs) (see (Goldberg, 1989) provide an alternative to traditional search techniques by simulating mechanisms found in genetics. Three notions are borrowed from biological systems:

the *phenotype*, which can be a living organism for biological systems or a design solution for design systems;

the *genotype*, which is a way of representing or encoding the information which is used to produce the phenotype; and

the *survival of the fittest*, which determines whether a genotype survives to reproduce.

In GA systems the genotype is usually represented as a binary string whose length varies with each application. For example, a genotype may look like: 001001101. The genotype representation allows combination or mutation to occur in order to construct better strings. Some measure of fitness is applied to each string after combination and mutation to determine which strings participate in generating the next generation of the population.

A simple genetic algorithm considers a population of n strings and applies the operators: reproduction (or selection), crossover, and mutation in order to create the next generation. Reproduction is a process in which strings are copied according to their fitness function. Crossover is a process in which the newly reproduced strings are mated at random and each pair of strings partially exchanges information. Mutation is the occasional random alteration of the value of one of the bits in a string.

In design case adaptation, the initial popoulation is the set of retrieved cases represented as genotypes. The attribute-value representation of cases is reinterpreted and used in the context of our genetic algorithm as follows. We consider the attributes used in the description of a case to be equivalent to genes. Thus, the collection of attributes used to describe a specific building corresponds to the building's genotype. The values that are associated with each attribute in the description of a case represent the structural or behavioural embodiment, in a specific building, of a general design feature. We thus consider the set of attribute values that make up a case description to be a phenotype.

Our genetic algorithm operates on individual buildings' genotypes by randomly mating and mutating them, detecting any changes in the corresponding phenotypes, determining if any of the resulting phenotypes is good enough to represent a solution to the problem being solved, and repeating the process if not. Each phenotype is a potential solution to the design problem being solved. New potential solutions to the problem are generated through the transformations of known phenotypes.

4.2 Shape emergence

Emergence is the process of making properties, which were previously only implicit in a representation, explicit. In the visual domain it is a common human process (Gottschaldt, 1926; Granovskaya et al, 1987). Figure 7 clearly demonstrates the phenomenon. If the right-hand figure is drawn using a CAD system, its representation will be that of six objects located in geometrical space. However, for humans the dominant features are the central star and triangles. None of the features seen by the human observer can be "seen", ie, are represented by the CAD system.

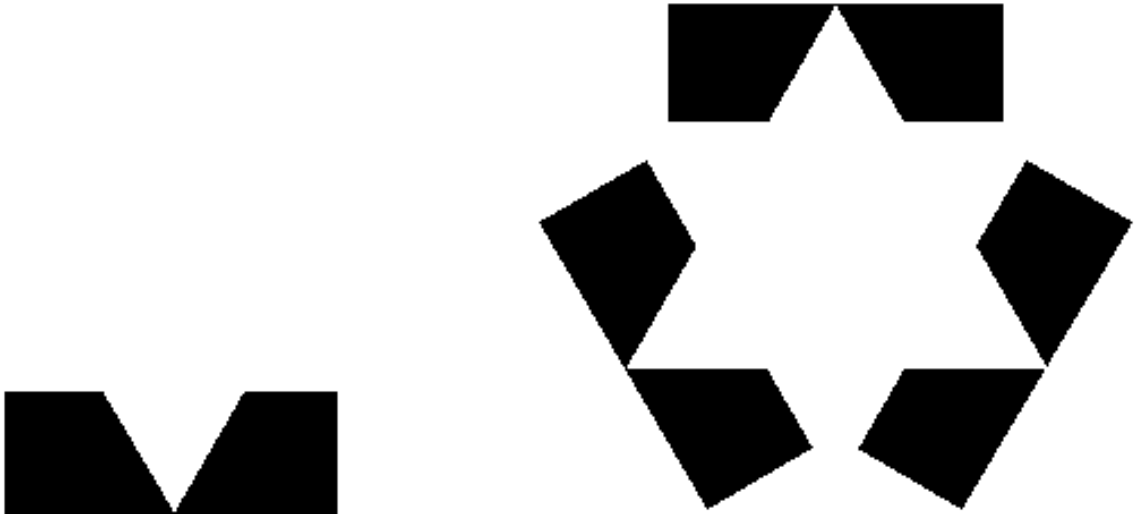


Figure 7. A object and a composite object, made of six copies of the single object, which exhibits strongly emergent shapes.

From the work of the Gestalt psychologists and more recently that of the cognitive psychologists, it is possible to construct computational models of shape emergence based on concepts drawn from their research. Humans appear to distinguish foreground from background in their reading of shapes. In order to emerge shapes which were not previously represented a process which manipulates the foreground and background can be constructed. What is done is to take the primary or originally represented shape and "unstructure" it so that it now becomes part of the background, producing an image composed of unstructured shapes only. A structuring process is then passed over this background to emerge foregrounds which may include both the primary shape and newly represented shapes. Gero and Yan (1993) have developed such a process based on a new representation, infinite maximal lines, along with a structuring process. Figure 8 outlines the overall process.

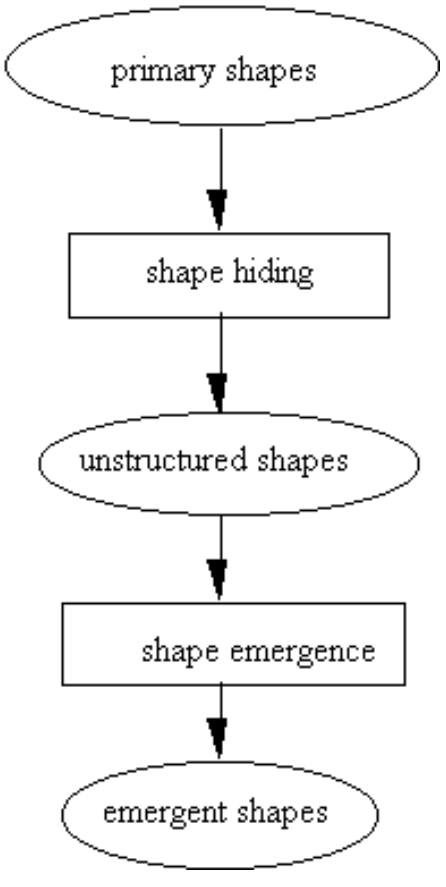


Figure 8. A process model of shape emergence (Gero and Yan, 1993).

The concepts behind shape emergence can be extended to emerge shape semantics, where the shape semantics are derived from visual patterns of shapes. Since these patterns were not originally represented they are emergent when there is a computational process which can find and represent them. From seeing drawings, various visual patterns are perceived by the human viewers. Designers can find different visual patterns from what was intended to be drawn. The newly discovered visual patterns may play a crucial role in developing further ideas in the same design if the designer is willing to adapt the visual pattern which was not there at the moment of drawing. Regardless of adaptability, visual patterns from shapes are defined as *shape semantics* when the patterns match the criteria for predefined labels, such as visual symmetry, visual rhythm, visual movement and visual balance. Figure 9 shows the facade of the Mosque of Djenne, in Mali. The human viewer can readily observe such shape semantics as reflectional symmetry, translational symmetry, balance and rythym in this facade.

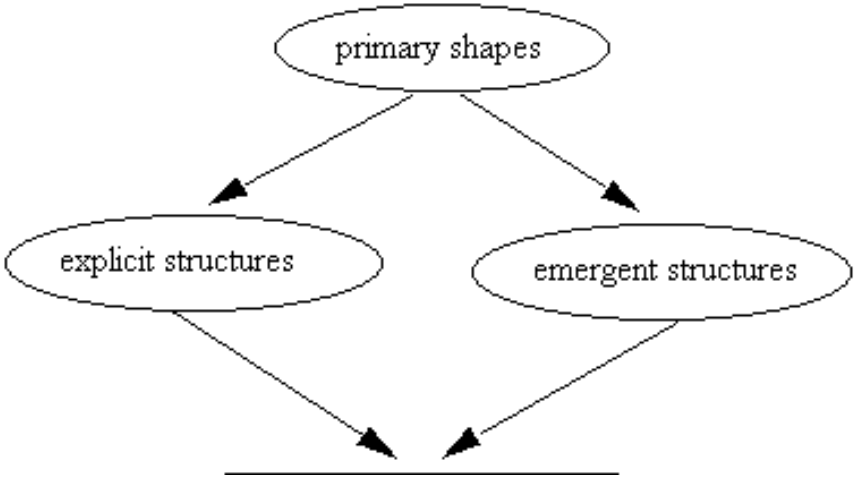
QuickTime™ and a
Photo - JPEG decompressor
are needed to see this picture

Figure 9: Various visual patterns can be perceived by the human viewer in the Mosque of Djenne, in Mali, rebuilt in 1905 (from Dethier, J., (1982), *Down to Earth*, Thames and Hudson, London, cover page).

Gero and Jun (1998) have developed a computational model of shape semantic emergence which is based on three processes:

- object correspondence
- grouping
- shape semantics emergence.

In order for shape semantics to exist there needs to be some form of structured regularity in the overall image. Object correspondence is the process which locates regularity of shape repetitions. Grouping locates regularity of groups of shapes, whilst the final process is hypothesis-driven and attempts to find known regularities amongst the groups of shapes, Figure 10.



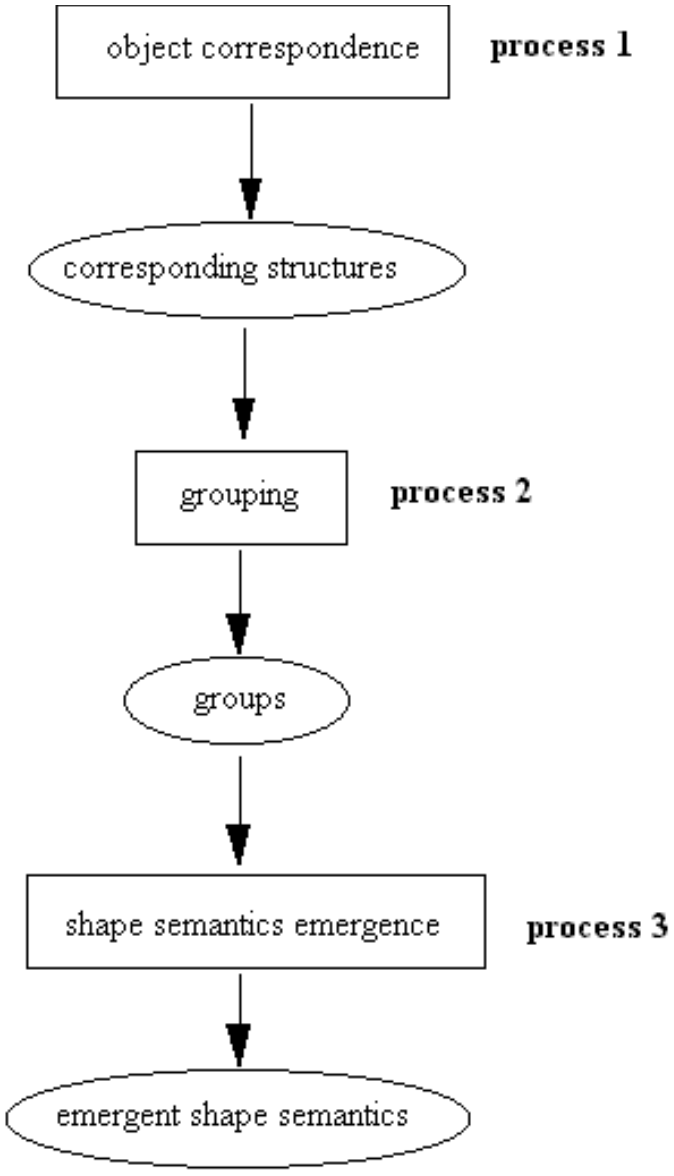
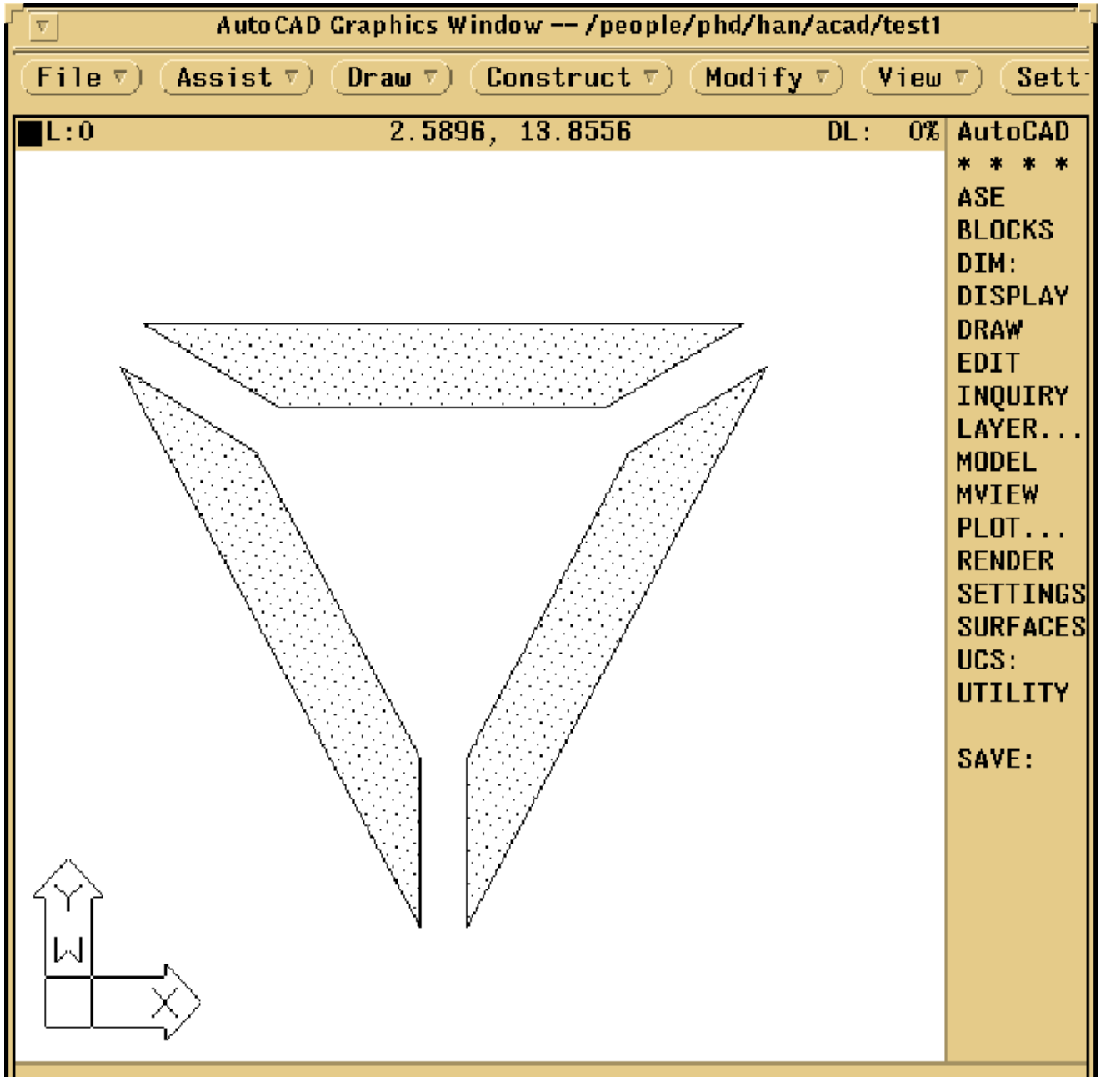


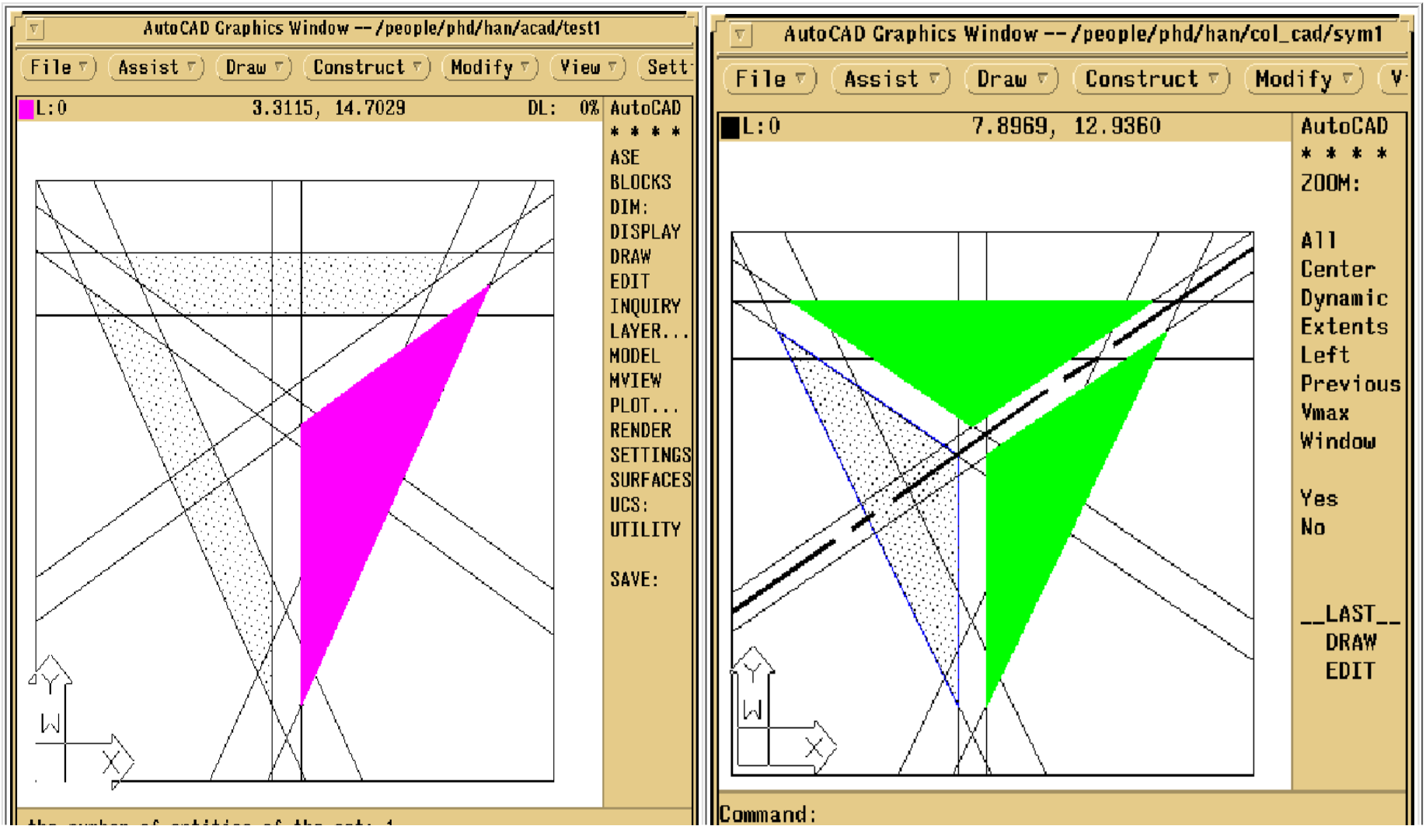
Figure 10. Process model for shape semantic emergence which includes shape emergence as a subprocess (Gero and Jun, 1998).

Figure 11 shows a screen dump of an implementation of the process outlined in Figure 10, the initally drawn image is shown. Figure 12(a) shows that an emergent shape has been found. Figure 12(b) shows that the shape semantic reflectional symmetry has been found.


AutoCAD Graphics Window -- /people/phd/han/acad/test1
File
Assist
Draw
Construct
Modify
View
Sett
L:0
2.5896, 13.8556
DL: 0%
AutoCAD
* * * *
ASE
BLOCKS
DIM:
DISPLAY
DRAW
EDIT
INQUIRY
LAYER...
MODEL
MVIEW
PLOT...
RENDER
SETTINGS
SURFACES
UCS:
UTILITY
SAVE:
Y
W
X

```
Command: open
AutoCAD Release 12 menu utilities loaded.
Command:
```

Figure 11. The primary shape as input in AutoCAD.



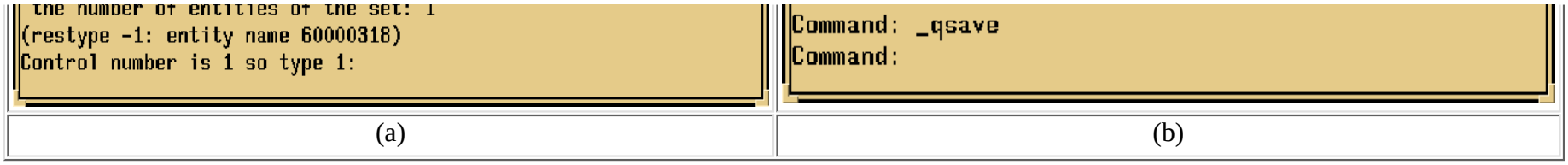


Figure 12. (a) Discovering emergent shapes; (b) discovering and using reflectional symmetry based on the emergent shapes (Jun and Gero, 1997).

Shape semantics play an important role in organising decisions, providing order, and generating final form in visually-oriented design. They appear to have a special role in architectural design in particular. Architecture reflects its main design concept through visual organization of structures. Visual organization of structures is shown as visual semantics of the design and is perceivable to designers. However, current computer-aided drawing, computer-aided drafting and computer-aided design systems prevent the discovery of visual shape semantics. Inadvertently such systems have enforced fixation so that it is not surprising that they are not used in the early stages of architectural design.

4.3 Design exploration as co-evolution

The search space for design is usually ill-defined and is accompanied by ill-defined goals. Hence, part of a design process is to *search* for the definition of the problem. Exploration has been defined verbally as "a problem is never final" (Logan and Smithers, 1993), "new dimensions are created during the process" (Gero, 1994) and "the design focus always changes" (Maher, 1994). The design problem, or what the designer is looking for, is reformulated in response to intermediate solutions, and co-evolves with the design solution.

A formal model of exploration has been developed, as illustrated in Figure 13, as the interaction of problem space and solution space (Maher and Poon, 1996). The problem space (or the functional requirements) is represented by *P*, and the solution space is represented by *S. Exploration* is defined as a phenomenon in design where *P* interacts and evolves with *S* over time.

This model of exploration has the following characteristics:

1. There are two distinct search spaces: Problem Space and Design Space.

2. These state spaces interact over a time spectrum.

3. Horizontal movement is an evolutionary process such that

a. Problem space P(t) evolves to P(t+1), P(t+2), and so on;

b. Solution space S(t) evolves to S(t+1), S(t+2), and so on.

4. Diagonal movement is a search process where goals lead to solution. This can be *"Problem leads to Solution"* (downward arrow) or *"Solution refocusses the Problem"* (upward arrow).

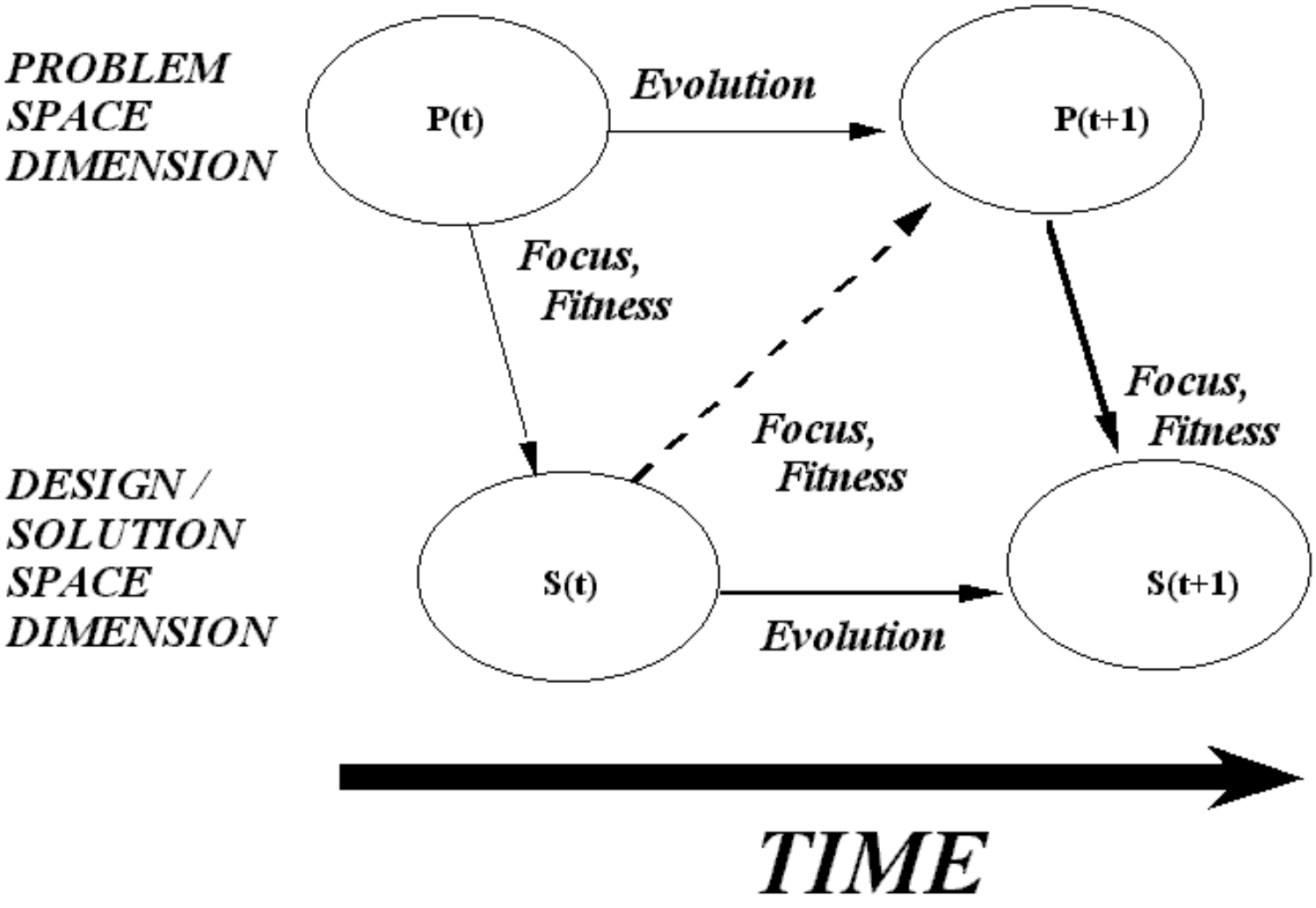


Figure 13. Problem-design exploration model.

The problem space P(t) is the design goal at time *t* and S(t) is the solution space which defines the current search space for design solutions. The solution space S(t) provides not only a state space where a design solution can be found, but it also prompts new requirements for P(t+1) which were not in the original problem space, P(t). This is represented by the dashed upward arrow from design space S(t) to problem space P(t+1). The upward arrow is an inverse operation where S(t) becomes the goal and a "*search*" is carried out in the problem space, P(t+1), for a "*solution*". This iterative relationship between problem space and design space evolves over time.

This model of exploration depicts an evolutionary system, or in fact, two evolutionary systems. The evolutionary systems are the problem space and the solution space. The evolution of each space is guided by the most recent population in the other space. This model is called co-evolution and provides the basis for a computational model of design exploration. The basis for co-evolution is the simple genetic algorithm where special consideration is given to the representation and application of the fitness function so that the problem definition can change in response to the current solution space.

5. Summary and Directions for Design Computing Research

This paper has described a framework within which design computing research is carried out. The three paradigms which have proven to be most useful are:

(i) empirically-based research (cognitive models);

(ii) axiom-based research (computational models); and

(iii) conjecture-based research (computational models).

(a) conjectures based on an analogy with cognitive processes; and

(b) conjectures based on an analogy with computational processes.

A number of research projects from the Key Centre of Design Computing, University of Sydney, have been presented as vehicles for each of these paradigms. Each of the projects uses one of the paradigms listed. The conduct of research for each of the projects is different and in some cases quite different. Empirically-based design computing research looks like experimental cognitive science research. Axiom-based design computing research looks like mathematical/logic research. Conjecture-based design computing research looks like theoretical engineering research. Thus, design computing research spans a range of research paradigms. What both the projects and the framework of paradigms imply is that design computing research has now reached a level of maturity that allows it to operate as a sub-discipline of design science rather than as simply a means of producing software packages. In this it contributes directly to the three goals enunciated in the Introduction. It is one of the primary means of developing theories, models and methods of designing as a process. It uses these theories, models and methods of designing as a process as a basis for the development of design tools, and is beginning to use the theories, models and methods as a basis for teaching (although this has not been presented in this paper).

What directions are open for design computing research? Not so much what projects should be pursued rather what strategic directions may yield results which inform us about designing and produce processes of value. As empirically-based research produces more results, we should have a greater understanding or how human designers design. Such knowledge will have implications for both how information technology can be interfaced with human designers and, perhaps more importantly, provide new conjectures for design computing research to explore in order to provide the foundation for more useful tools for designers. Similarly, as the other approaches yield insights into designing they may provide the foundation for novel tools.

References

Damski, J. C. and Gero, J. S. (1996) A logic-based framework for shape representation, *Computer-Aided Design* **28**(3):169-181.

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

Gero, J. S. (1994). Towards a model of exploration in computer-aided design, *in* J. S. Gero and E. Tyugu (eds), *Formal Design Methods for Computer-Aided Design*, North-Holland, Amsterdam, pp. 271-291.

Gero, J. S. and McNeill, T. (1997). An approach to the analysis of design protocols, *Design Studies* (to appear).

Gero, J. S. and Jun, H. (1998) Emergence of shape semantics of architectural shapes, *Environment and Planning B: Planning and* ***Design*** (to appear).

Gero, J. S. and Yan, M. (1993). Discovering emergent shapes using a data-driven symbolic model, *in* U. Flemming and S. Van Wyk (eds), *CAAD Futures'93,* North-Holland, Amsterdam, pp. 3-17.

Goldberg, D.E., (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading.

Gottschaldt, K. (1926). "Uber den Einfluss der Erfahrung auf die Wahrnehmung von Figuren." *Psychologische Forschung* No. 8, pp. 261-317.

Granovskaya, R. M., Bereznaya, I. Y. and Grigorieva, A. N. (1987). *Perception of Forms and Forms of Perception*, Lawrence Erlbaum, Hillsdale, New Jersey.

Gómez de Silva Garza, A. and Maher, M.L. (1996). Design by interactive exploration using memory-based techniques, *Knowledge-Based Systems*, **9**(1):151-161.

Jun, H. and Gero, J. S. (1997). Representation, re-representation and emergence in collaborative computer-aided design, *in* Maher, M. L., Gero, J. S. and Sudweeks, F. (eds), *Preprints Formal Aspects of Collaborative Computer-Aided Design,* Key Centre of Design Computing, University of Sydney, Sydney, pp.303-319.

Logan, B. and Smithers, T. (1993). Creativity and design as exploration, *in* J. S. Gero and M. L. Maher (eds), *Modelling Creativity and Knowledge-Based Creative Design, Lawrence*, Erlbaum Associates, pp.139-175.

Maher, M.L. (1994). Creative design using a genetic algorithm, *Computing in Civil Engineering,* American Society of Civil Engineers, pp 2014-2021.

Maher, M.L., Balachandran, B. and Zhang, D.M., (1995). *Case-Based Reasoning in Design*, Lawrence Erlbaum, Hillsdale, New Jersey.

Maher, M.L.and Balachandran, B. (1994). Multimedia approach to case-based structural design, *Journal of Computing in Civil Engineering* **8**(3): 359-376.

Maher, M.L. and Poon, J. (1996). Modelling design exploration as co-evolution, *Microcomputers in Civil Engineering* **11**:195-210.

Maher, M.L., Simoff, S. and Cicognani, A. (1997). Observations from an experimental study of computer-mediated collaborative design, *in* M. L. Maher, J. S. Gero, and F. Sudweeks (eds*). Preprints Formal Aspects of Collaborative CAD,* Key Centre of Design Computing, University of Sydney, Sydney, pp.165-185.