INTERNATIONAL DESIGN CONFERENCE - DESIGN 2004
Dubrovnik, May 18 - 21, 2004.

DESIGN 2004

SITUATED DESIGN COMPUTING: INTRODUCTION AND IMPLICATIONS

J. S. Gero

Keywords: design computing

1. Design Computing

Design computing is the area that deals with the development and use of computers in design. Both as a research field and an application field it has a long history stretching back to the early 1960s when the focus was on layout planning. There has been a number of stages in the development of design computing. Whilst each stage built on the previous stage, subsequent stages did not necessarily subsume the previous stages, which often maintained their own separate activity.

The first clearly defined stage was concerned with simulation. Here the focus was on developing algorithms for the visual and environmental analysis of designs. This immediately brought with it the need to represent buildings, along with the need to have data structures and graphical display representations. Simulation continues to remain a major research and application focus to this day. However, the emphasis has changed from the analysis of performance in individual building behaviours to a synthetic approach built around life cycle and sustainability approaches.

The next stage was concerned with computer-aided drafting (CAD) where the focus shifted increasingly to data structures and computer graphics and then later databases. Whereas the first stage was closely linked to buildings and architecture, this second stage drew more from computer science developments in computer graphics, data structures and databases.

The next stage took computer-aided drafting and expanded it to become computer-aided modeling. Again, this built on developments in computer science, this time on developments in object modeling. Computer-aided modeling has grown to encompass computer-aided representations generally including multimedia modeling.

The access to the internet through the World Wide Web opened up new possibilities in design computing. These have resulted in collaboration across the net through virtual design studios and the creation of virtual environments. The latter opened up new areas of design beyond the use of the virtual as a simulation of the physical by making the design of virtual environments a design field itself.

Paralleling these developments has been a constant interest in computational support across a wide range of design processes beyond those of representation. Whilst there has been considerable uptake of design synthesis support in various engineering disciplines the same cannot be said for architecture. The focus of design support continues to be in the representation, modeling and direct analysis areas.

2. Some Problems with Current Design Computing

Current design computing is a direct outgrowth of computer science and much of it is based on similar concepts to those on which traditional artificial intelligence is based. These include fundamental notions that are themselves drawn from the physical sciences, namely, that the world is there to be modeled and is unchanging except in formally predictable ways. Further implicit assumptions include:

- it doesn't matter where the computation is done, you will always produce the same result,
- it doesn't matter when you do the computation, you will always produce the same result,
- it doesn't matter 'who' does the computation,
- results are independent of the expectations of the computational system,
- memory access is independent of its use, and
- memory access in independent of other memories.

This is a partial list of the some of the issues that current design computing does not address. Current systems are developed unaware of these assumptions and therefore have no direct capacity to address them.

3. Situated Design Computing

Situated computing takes concepts from cognitive science to address some of the assumptions. The foundations of situated design computing are presented in the next section. Here we present in visual form some of the issues that need to be addressed. Here are some examples.

3.1 Where you are when matters

In design computing there is no notion that it matters where a system is when it does what it does. Take as an example, a visual recognition system. If it were exposed to the image in Figure 1 and then later to the image in Figure 2, it would not be able to construct the image that can be emerged in Figure 3 unless it saw both the images in Figures 1 and 2 at the same time.

Figure 1. First image seen

Figure 2. Second image seen

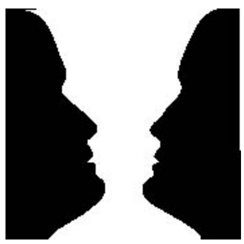


Figure 3. Only when both images are seen at the same time can the white image of a vase emerge: where you are when matters

Time plays no role in current representations. As a consequence those aspects of human understanding that are based on time are not able to be dealt with.

3.2 The external world is not there to be modelled

A common assumption is that the external world is there to be represented, ie that in some sense it has only one representation. This misses an important step: namely that of interpretation. Before anything can be represented it needs first to be interpreted and it is that interpretation that is represented. Consider the image in Figure 4. Is it a set of triangular shaped objects pointing towards to top right with their bases faces downwards to the left? Or is it a set of triangular shaped objects pointing to the top left with their bases pointing to bottom right? Or is it as et of triangular shaped objects pointing downwards with their bases pointing upwards? Each of these interpretations is unique and is not simply a rotation of another interpretation. There are many other interpretations possible.

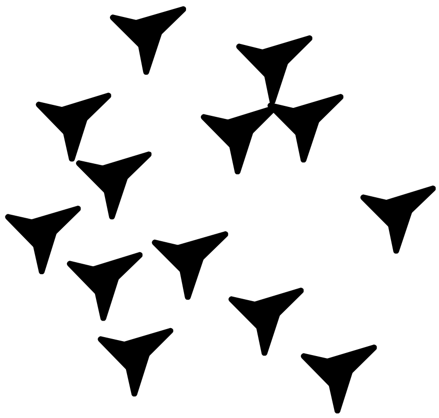
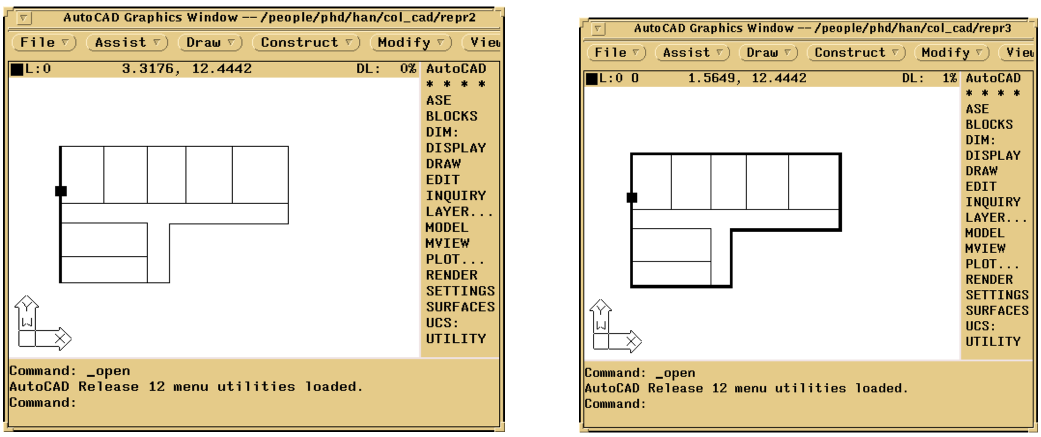


Figure 4. The external world needs to be interpreted before it can be represented

This fundamental difference between image and its representation is well known in computer-aided modeling systems. Consider the two screen shots in Figures 5(a) and 5(b). Both images are the same but their underlying representations are not evident in the image. An indication of the difference can be found by picking the same line (indicated by the black dot on the left-hand edge of the image) in both images. In Figure 5(a) the entire perimeter is highlighted indicating that a single polyline was used to represent the perimeter. In Figure 5(b) the same pick highlights the left hand edge of the perimeter only, indicating that the perimeter is represented as line segments.



(a) **(b)**

Figure 5. The same image has different representations that depend on the individuals who created them rather than on any objective knowledge

3.3 When you do a computation matters

Typically, in a standard design computing system the order in which you carry out computations that utilizes the memory of the system makes no difference. However, there are many examples where we would expect the order to make a difference. Take as an example the access to a new telephone number. We would expect that the first access to a new number should take longer than after it has been accessed a number of times. If the number is accessed very rarely then we accept that it may take longer to find than if it is in regular use.

A number of approaches partially similar to this based on chronological recency have been implemented, for example the RETE algorithm (Forgy 1982). However, none of these approaches has any universality.

4. Foundations of Situated Design Computing

Situated design computing take two foundational concepts from situated cognition (Clancey 1997). The first is called constructive memory and the second situatedness.

4.1 Constructive memory

Constructive memory is based on the notion of memory that is fundamentally different to current views of computational memory. Current computational memory is a thing in a location. The thing is generally a value of a variable and that value is unchanged by just accessing it. The location is fixed and is independent of its content and use. Everything about the memory is independent of its use.

Constructive memory takes the view that memory is made up of sensate experiences, which are conceptually close the current views of computationally memory, and memories that are constructed on the demand to have a memory (Rosenfield 1988). If there is no demand for such a memory then no memory is constructed and the system has no memory of that. Thus, memory becomes a process that produces results that are similar to experiences. These newly constructed "memories" are then available to be used to construct memories when there is a demand for another memory that is not a previous experience.

The process of memory construction, Figure 6, commences with a demand for a memory that is the cue. This cue is used to bring together experiences that along with knowledge related to the situation construct the memory. In doing so the experiences that were used in the construction of this memory change their likelihood of being used again. In Figure 6 this is indicated by their "level of processing". The effect of this is that experiences that have been used often to construct memories are increasingly likely to play a role in the construction of future memories. This changes the way existing experiences are themselves stored.

An elementary example of this process in humans is the way a new telephone number becomes easier to recall as it is given out to many people and hence accessed more often. A richer example relates to when a person sees a movie and forms an opinion of it but their opinion changes as they hear other people discussing the movie. Their memory of the movie is no longer based on their own sensate experience but is a construction that relates to their interaction with others.

Computationally there are a number of possible implementations of this idea of constructive memory. One question that needs to be answered is whether to treat constructed memories in the same way that experiences are treated, ie sensate experiences and constructed memories become indistinguishable. If they are treated differently then in what way should they be different? The answers to this and related questions define the specific architecture of the memory system.

If we allow constructed memories to be treated as experiences the system can no longer distinguish between what it experienced and what it constructed as a memory but now treats as an experience. Figure 7 shows the case where a constructed memory is treated as an experience. A new memory is demanded and this demand is the cue to construct a memory. The same experience as was used recently is utilized in constructing that new memory. As a consequence that experience's level of processing is raised further, thus increasing its likelihood of being used again.

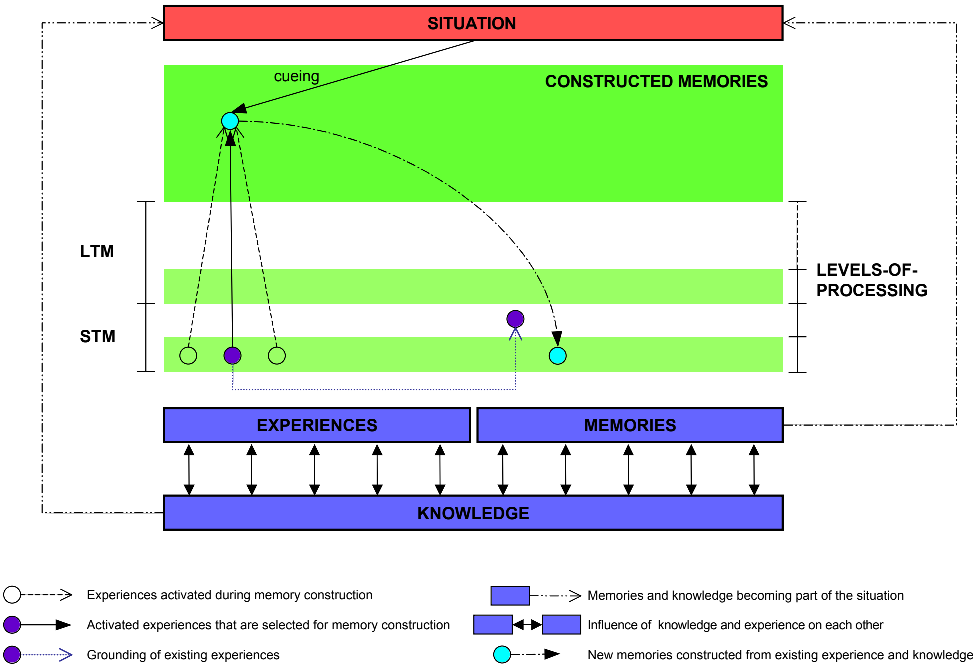


Figure 6. Memory is constructed not retrieved. A demand for a memory is used as a cue (Liew and Gero 2004)

This form of memory can be viewed as an experience-based memory system that is developed through the interaction of the system and its environment. Take as an example a vision system in a design environment that is used to categorize shapes. If it has capabilities to categorize arbitrary shapes but is only and consistently exposed to rectilinear shapes such that its memory is only composed of rectilinear shapes then when it is later exposed to circular shapes it will have difficulty categorizing those shapes compared to categorizing a rectilinear shape.

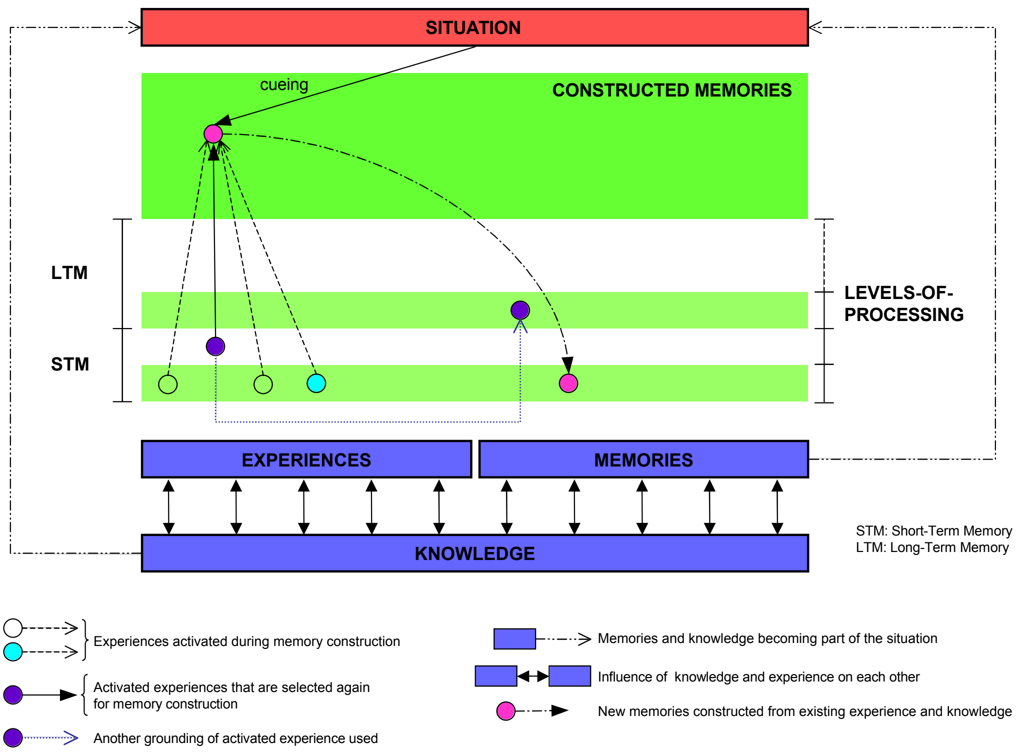


Figure 7. As memory is constructed the experiences that contribute its construction change the likelihood of their being used in the future (Liew and Gero 2004)

There is a very important effect as a consequence of this memory construction process. All experiences and memories that have been constructed are always only “seen through the lens of the present”. This means that all new memories are just-in-time memories that relate to the current situation and not simply recalls of previous experiences. Constructive memories are a form of learning where each new memory changes what has been learnt before.

4.2 Situatedness

Situatedness is the framework for constructive memory. It refers to the notion that a computational system operates within a world of its own interpretation rather than simply within a predefined environment. This world is based in part on the knowledge developed through the interaction between the system and its environment rather than completely on an encoding of the knowledge needed for any interaction. The fundamental concept here is that the world is not there to be modeled but rather to be interpreted before it can be modeled. This interpretation is partly a function of the expectation of the system. The same world can be interpreted differently depending on the expectations of the observer. We are familiar with this in the construction industry where the architect and the engineer observing the same drawing see very different issues.

Figure 8 shows a conceptual architecture of a computational system based on situatedness. The system interprets its external world to produce its internal world based on its own experiences. This interpreted world is used to hypothesize an expected world. The three world are defined as follows:

- *external world*: the world that is composed of representations outside the designer or design agent.
- *interpreted world*: the world that is built up inside the designer or design agent 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.
- *expected world*: 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.

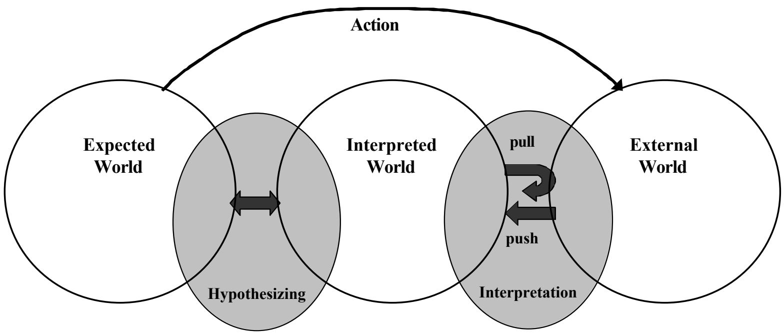


Figure 8. Situatedness in designing involves the interactions between three worlds (Gero and Kannengiesser 2003)

These three worlds are recursively linked together by three classes of processes. The process of *interpretation* transforms variables, which are sensed in the external world into the interpretations of sensory experiences, percepts, and concepts that compose the interpreted world. This is done by the interaction of sensation, perception and conception processes (Gero and Fujii 2000). The process of *focussing* focuses on 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. The process of *action* is an effect, which brings about a change in the external world according to the goals in the expected world.

Situated design computing has the capacity to be the basis of computational models of designing that more closely account for the observed behaviour of designers. It has the capacity to model the changes in interest of the designer as he/she observes emergent structures in their designs, structures that could not have been predicted at the outset. Such emergent structures map onto the concepts of “where you

are when you do what you do matters” that is one of the bases of the interaction process in constructive memory.
This approach to computational models and support for designing does not require the designer to take a particular path through any series of designing activities. It is the antithesis of planning in the artificial intelligence sense. It allows the designer or a design system to change their trajectory and does not require either monotonicity or logical consistency. Using the notion of constructive memory it need make sense only retrospectively.
Dewey quoted by Clancey (1997) summarizes this idea succinctly:

> “Sequences of acts are composed such that subsequent experiences categorize and hence give meaning to what was experienced before.”

5. Examples of Situated Design Computing

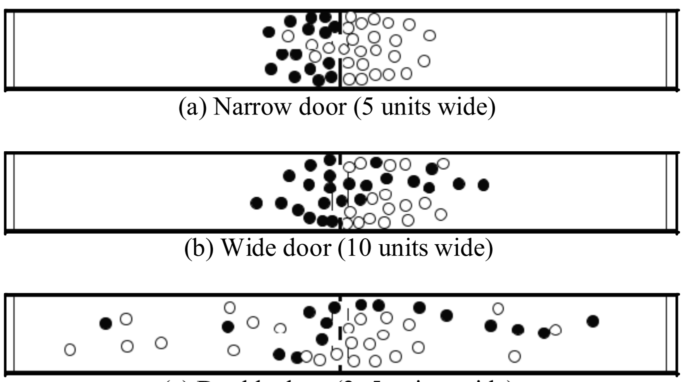
5.1 Simple situated pedestrians in door simulation

One way to simulate the behaviour of a door is to build a queuing model where the door is treated as a server and the pedestrians as units to be served with a defined stochastic arrival rate. This type of modeling has a high granularity associated with it and does not allow for the observation of any emergent behaviour. An alternate approach is to use simple situated agents as the pedestrians and to allow them to interact with each other and the door and observe the behaviour.
Here the pedestrians are simple situated agents. Each pedestrian uses the same knowledge about its interaction with its environment. This knowledge is derived from the social force model (Helbing and Molnár 1995) and covers the four areas of efficiency of movement, distance from other pedestrians, distance from obstacles and attraction to other pedestrians, presented in Table 1. These are sufficient to develop a situated simulation based on the social force model.

Table 1. The four classes of knowledge needed for the social force model of pedestrian behaviour

	Description of social force
1.	Pedestrians are motivated to move as efficiently as possible to a destination.
2.	Pedestrians wish to maintain a comfortable distance from other pedestrians.
3.	Pedestrians wish to maintain a comfortable distance from obstacles like walls.
4.	Pedestrians may be attracted to other pedestrians (e.g. family) or objects (e.g. posters).

Figure 9 shows the same situated agents in three different environments. Their behaviour in response to their interaction with their environment varies as can be seen in Figures 9(a), 9(b) and 9(c).



(a) Narrow door (5 units wide)

(b) Wide door (10 units wide)

(c) Double door (2x5 unites wide)

Figure 9. Screenshots of the simulations of pedestrian flow through (a) a narrow, (b) a wide, and (c) a double doorway design with a crowd of 40 pedestrians. The black circles indicate pedestrians traveling from left-to-right across the doorway and the white circles indicate pedestrians moving from right-to-left (Saunders 2001)

Very clear emergent patterns of behaviour can be observed in the screenshots. When the door is too narrow there is congestion. When there are two doors the pedestrians organize themselves so as to move effectively and efficiently by each direction largely moving through one door only. This behaviour is not programmed into the system but is a consequence of the interactions of the pedestrians with their environment that includes each other.

5.2 Curious Situated Agents

We can build situated agents using the architecture outlined in Gero and Fujii (2000). This architecture, Figure 10, is based on a cognitive approach and uses the processes of sensation (S in Figure 10) to bring in data from the external world, perception (P) to structure sensate data and to bias sensation, conception (C) to generalize percepts driving actions (A) and finally effectors (E) on the environment (e). The architecture includes both short-term “memory” (STM) and long-term “memory” (LTM). .

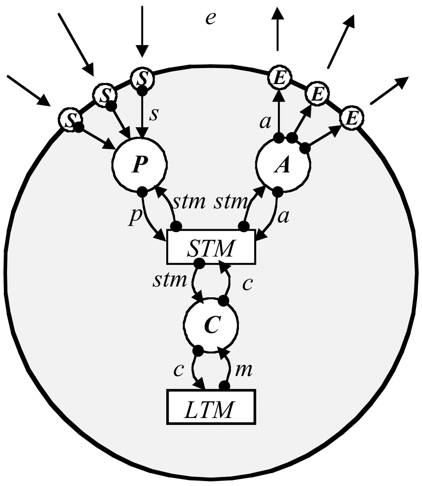


Figure 10. The architecture of a situated design agent

This architecture forms the basis of a number of developments including the addition of curiosity that is used as a motivating force for all an agent’s activities. This involves the addition of novelty (N) and interestingness (I) to produce curiosity (X) as shown in Figure 11 (Saunders and Gero 2001).

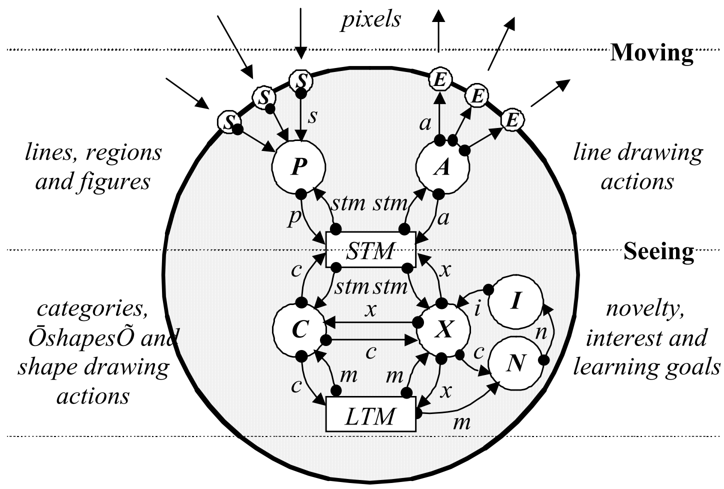


Figure 11. The architecture of a situated design agent with curiosity exemplified within a drawing environment (Saunders and Gero 2001)

Curious situated agents have been the basis of various applications including situated sketching where the agent develops knowledge about shapes from its own sketching activities (Saunders 2001) and situated evaluations where agents are used to assist in the curating of an art exhibition (Saunders and Gero 2004).

6. Implications of Situated Design Computing

The foundational concepts of situated design computing have been briefly introduced in this paper. They can be summarized as:

- memory construction not recall
- interaction not encoding.

The effect of the use of situatedness in design computing is not to overthrow or even necessarily displace existing approaches, rather it is to augment existing approaches with some possible displacement. It will allow the use of computational support in areas where current implementations are not suitable because of the rigidity of that approach.

One area that situated design computing has implications is that of adaptive tools. It is claimed that situated agents that utilize constructive memory are capable of both the learning and adaptation needed to allow tools to continuously adapt themselves to their use. Constructive memory systems have been developed that have behaviors approaching those needed for tool adaptation (Liew and Gero 2002). We are now in a position to bring these two concepts together to produce design tools as situated agents that adapt to their use. This would produce a new generation of design tools, tools that change their behavior based on their interaction with their users. This has the potential to produce personalized versions of tools.

A number of new areas open up as a consequence of these ideas. Optimization has been a powerful approach in engineering design but has been limited by the need to predefine all the variables and their relationships a priori and to fix the objectives. A situated approach to design optimization would allow the restructuring of design generators as a result of the interactions that occurred in getting to the current state. As an example of this, consider the case where beneficial emergent structure properties can be found in a population of potential solutions. These properties can be reverse engineered into the design generator to produce them as part of its execution.

New ways of studying design creativity open as when we focus on the situated interactions between players in a social system that includes "creative" designers. This moves the study of creativity away from a unique focus on individuals and onto the social interactions between them (Sosa and Gero 2004).

Acknowledgements

This work has been supported by a University of Sydney Sesqui Research and Development grant. The ideas have benefited from discussions with Rob Saunders, Udo Kannengiesser, Greg Smith and PakSan Liew.

References

Clancey, W 1997, Situated Cognition, Cambridge University Press, Cambridge.

Forgy, C: Rete: A fast algorithm for the many pattern/many object pattern match problem, Artificial Intelligence, 19(1982) 17-37.

Gero, JS and Fujii, H: 2000, A computational framework for concept formation in a situated design agent, Knowledge-Based Systems 13(6): 361-368.

Gero, JS and Kannengiesser, U: 2003, The situated Function-Behaviour-Structure framework, Design Studies (to appear)

Helbing, D and Molnár, P: 1995, Social force model for pedestrian dynamics, Physical Review E, 51: 4282–4286.

Liew, P and Gero, JS: 2002, An implementation model of constructive memory for a situated design agent, in JS Gero and . Brazier (eds), Agents in Design 2002, Key Centre of Design Computing and Cognition, University of Sydney, Australia, pp.257-276.

Liew, P and Gero JS: 2004, Constructive memory for situated agents, AIEDAM (to appear).

Rosenfield, I: 1988, The Invention of Memory, Basic Books, New York.

Saunders, R: 2001, Curious Design Agents and Artificial Creativity, PhD Dissertation, University of Sydney.
Saunders, R and Gero, JS: 2001, Designing for interest and novelty: Motivating design agents, in B de Vries, J van Leeuwen and H Achten (eds), CAADFutures 2001, Kluwer, Dordrecht, pp.725-73.
Saunders, R and Gero, JS: 2004, Curious agents and situated evaluations, AIEDAM (to appear)
Sosa, R and Gero, JS: 2004, A computational framework to investigate creativity and innovation in design, in JS Gero (ed), Design Computing and Cogition'04, Kluwer, Dordrecht (to appear).

John S. Gero, Prof. Dr.
Univeristy of Sydney, Department of Architectural and Design Science
Key Centre of Design Computing and Cognition
E-mail: john@arch.usyd.edu.au