



# Situated Design Computing: Principles

**J.S. Gero**

**Krasnow Institute for Advanced Study**

**Volgenau School of Information Technology and Engineering**

**George Mason University**

**Fairfax VA, United States of America**

## Abstract

This paper introduces the principles of situated design computing. Situated design computing expands the concept of computing from being the encoding of objective knowledge that is fixed by the programmers of the system to include first-person knowledge acquired while the programs are being used. This first-person knowledge forms the basis of the system's experience which is then made available as the system is used further. The four principles of situatedness are presented along with the basic concept of the mechanism by which such systems operate: constructive memory.

**Keywords:** design computing, situatedness, first-person knowledge, constructive memory.

## 1 Introduction

Computational design tools aim at encoding knowledge and making it available in an objective manner to the designer. There is an assumption behind this approach, namely that all the knowledge is objective, *ie*, independent of the user. Examples of objective knowledge include stress analysis, determining the position of the sun, thermal analysis of a building and methods such as linear programming in optimisation. Such objective knowledge tends to be deductive in nature. The most powerful examples of deductive knowledge are those based on axioms from which subsequent theorems have been developed. These theorems map onto the behaviour of the world. In addition, there is a category of knowledge that is based on induction (*ie*, knowledge learned from examples without causality). Even inductive knowledge is treated as objective, *ie*, independent of the user. For example, a layout algorithm that utilises some heuristics is used as if there were causality encoded. This has served design computing well. It has allowed for the widespread distribution of computational tools. It has provided the basis of transferable skill development in

users. As our knowledge of the world has improved, so we have been able to update the knowledge in these programs.

This paper develops an approach that aims to extend our understanding of what kinds of knowledge we can expect our computational tools to have and how systems that have a range of kinds of knowledge might perform differently.

## 2 First-Person Interaction vs Deductive Knowledge

Whilst it is clear that much of human knowledge is objective, in the sense described here, there is a category of everyday knowledge that depends on the person rather than deduction. This knowledge is developed based on first-person interaction with the world. This class of knowledge is sometimes inappropriately encoded as deductive knowledge and in so doing often causes a mismatch between the experience of the person who encoded the knowledge and a subsequent user of that knowledge.

A simple example of such encoding of personal knowledge can be seen even in the way objects are represented in a CAD system. Figure 1(a) shows the screen image of a floor layout. Simply looking at the drawing of the floor layout gives no indication of how it has been encoded. The darkened line is the single polyline representation of the outline obtained by pointing to a spot on the boundary, but that representation could not be discerned from the image. Figure 1(b) shows exactly the same outline but it is encoded differently, as indicated by the darkened polyline obtained by pointing to the same spot.

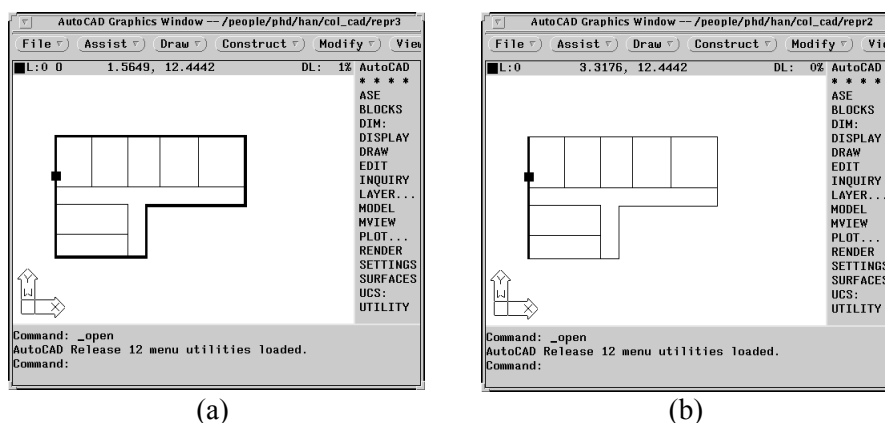


Figure 1. The same image has different encodings (a) and (b) that depend on the individuals who created them rather than on any objective knowledge

The issue here is one of interpretation that has been missing in design computing. 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 this interpretation that is represented. This is an example of first-person interaction with a design.**

Consider the image in Figure 2. Is it a set of triangular shaped objects pointing towards to top right with their bases facing 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 a set 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.

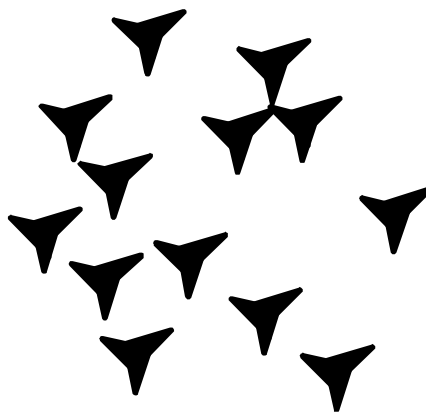


Figure 2: The external world needs to be interpreted before it can be represented.

The direction you perceive the objects, in Figure 2, to be moving is a function of the representation: if representation has the bases pointing to the bottom left of the figure, then you will see them moving towards the top right; however, if the representation has the bases pointing upwards, then the objects appear to be moving downwards.

A coherent and unified development that includes the acquisition and use of this first-person knowledge in a form that can be readily understood, implemented and tested is currently under development in design computing. That lack of the availability of first-person knowledge has prevented design computing from being used further to develop new classes of tools to support design. The field that deals with this is called *situated design computing* and the reasoning processes are called *situated reasoning*.

The research for situated design computing is founded on notions from:

- situatedness, which is concerned with being in a particular place at a particular time and how the world is viewed from that place at that time [1],
- situated cognition in cognitive science, which is concerned with the cognitive structures and conceptual coordination to deal with situatedness [2], [3], [4] and [5],

- research related to first-person interactions with representations, which includes the expectations of the person or system carrying out the interpretation [6], [7] and [8], and
- constructivist memory, which is concerned with memory as a process to construct a memory cued on a demand to have such a memory rather than a recall of elements in a location [9], [10], [11] and [12].

### 3 Designing

Designing can be characterised 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” [13]. During designing, a designer determines the variables that contribute towards the function, behaviour and the structure of the design according to his/her experiences, knowledge and conception of what is presented to him/her (constructed as the situation). These behaviour and structure variables are not chosen *a priori* but are produced in response to the various situations as they are constructed by the designer. What the designer has done previously, both prior to this design and during the current design process affects how the designer constructs the situation and what memories he/she brings to bear on the current situation [4].

Much of human design knowledge is individual and based on the first-person interaction with the design as it develops biased by the designer's previous experiences. This is one of the reasons why different designers confronted with the same requirements produce different designs. If designers all produced the same design then we would have catalogues of designs and not require designers to work on each problem separately. The implication of this is that first-person design knowledge cannot be encoded directly into a tool by the tool builder but can only be developed through the experience of doing.

**The inability of the current design computing paradigm to encompass first-person knowledge has limited the applicability of computers to a restricted class of applications where only objective knowledge plays a role.**

When these systems have been used where first-person knowledge is needed they have failed to produce the efficacy expected because all their knowledge is hard-coded. Another computational paradigm is needed to augment the current objective knowledge-centred one to increase the applicability and effectiveness of computational support for design.

Human designers work within a world that is based on their perceptions. Although much of the world is defined for them in the way it is presented to them, it depends on what the designer thinks is significant, as that changes what they think the design issues are and what sorts of means are open to them to design solutions to these problems. This perceived world is an interpretation by the designer of the

world external to them biased by their own experiences and knowledge. The designer then produces an expected world as a result of designing. When the expected world produces actions on the external world then a design is realized. This interaction between the designer and the environment strongly determines the course of designing and the resulting worlds and is called the *situation*, whose foundational concepts go back to the work of Dewey [9] and Bartlett [14]. In paraphrasing Clancey [2] we can summarize it as “where you are when you do what you do matters”. Figure 3 shows the relationship between these worlds.

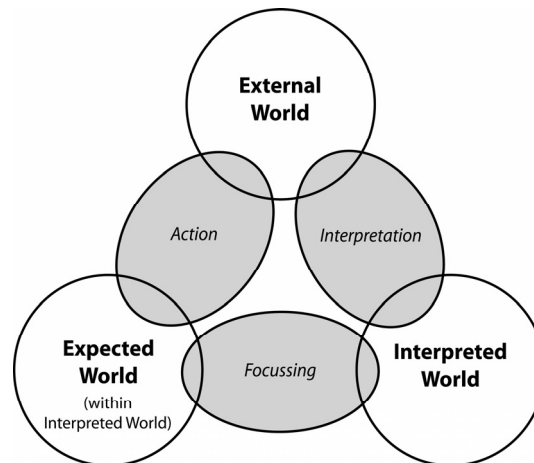


Figure 3: The *situation* arises from the designer’s interactions with the external, internal and expected worlds [15]

In the remainder of the paper we will examine a set of principles that is used to develop a computational paradigm that supports first-person knowledge and discuss some issues and effects that follow from this.

## 4 Principles of Situatedness

This section introduces the four principles and one lemma of situatedness.

### 4.1 Principle of Effect: *What a System Does Matters*

The implication of this principle is that actions produce effects in terms of the production of first-person knowledge. In traditional uses of computers in designing, the computational systems are unchanged by their use. This makes sense if the system only embodies third-person knowledge. However, the use of any system, whatever kind of knowledge it contains, generates first-person knowledge in the system that uses it. This is the first fundamental distinction between traditional design computing and situated design computing. For example, an optimization approach used in the design of layouts could produce first-person knowledge in the form of successful strategies. These could then be used next time the optimization program was used for layouts.

## 4.2 Principle of Nonlinear Temporality: *When a System Does What It Does Matters*

In traditional design computing when you carry out a computation it plays no role in the result that is produced. Again, this makes sense if the system only embodies third-person knowledge. However, if the system embodies first-person and generates first-person-knowledge as it runs, the chronology of the system's use affects what is used and what is learned. As the boundary condition for this principle consider that if A is carried out before B, then A cannot make use of any knowledge acquired through the execution of B, but B can make use of knowledge acquired through the execution of A.

## 4.3 Lemma of Experience: *What a System Did Before Affects What It Does Now*

As a consequence of principles 1 and 2, we can state this lemma: What a system did before affects what it does now. This is one definition of experience. Experience has the potential to guide future actions. The effect of this is that situated design computing systems are not static systems but are dynamic in terms of their behaviour. This concept can be applied recursively, so that previous experiences are used to produce current experiences.

## 4.4 Principle of Locality: *Where a System Is When It Does What It Does Matters*

First-person knowledge includes not only what happened and when it happened but also where it happened. The system has to be at the "right" place at the "right" time for unique events to occur. As an example consider a system that is exposed first to the image in Figure 3(a) and then later to that in Figure 3(b). These two exposures are insufficient to produce the emergence that is possible if the system is at a location that allows it to be exposed to the image in Figure 3(c).

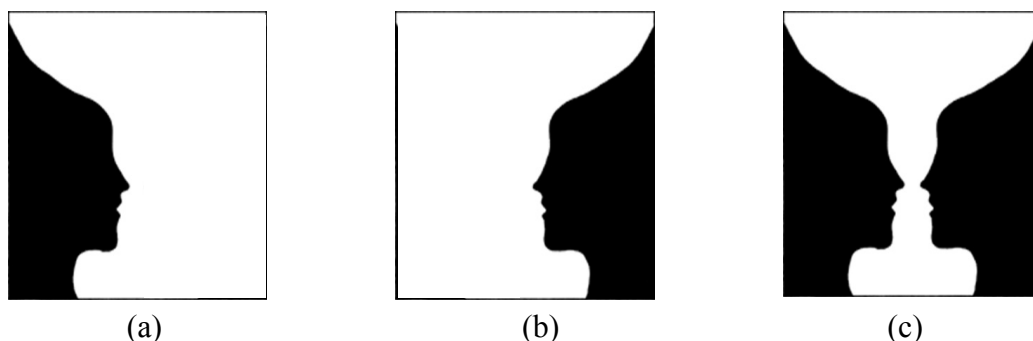


Figure 3. The potential for emergence exists only if a system is where the image in (c) is. It is insufficient to be exposed to the images in (a) and (b) at different times to produce the emergence that can be produced if the system is exposed to (c).

## 4.5 Principle of Interaction: *Who and What a System Interacts With Matters*

Interaction is one of the distinguishing characteristics of a situated design computing system. Without interaction there is no potential to produce first-person knowledge. There are two sources of interaction for a system: other computer programs and users of the system. Conceptually there is no difference between them in terms of interactions. Each interaction has the capacity to use previous experiences and to produce first-person knowledge. Interactions have the potential to change the meanings of experiences.

## 5 Constructive Memory

How can we build systems with these capabilities? We need the capacity to develop first-person knowledge and to make it available in a useful form whenever it is potentially needed. We can take ideas from Barlett's work on memory [14].

Computationally memory has come to mean a thing in a location. The thing can take any form and the location need not be explicitly known. The thing can be accessed by knowing either its location or its content. There are a number of distinguishing characteristics of this form of memory:

- memory is a recall process
- there needs to be an explicit index (either location or content)
- the index is unchanged by its use
- the content is unchanged by its use
- the memory structure is unchanged by its use.

This is in contrast to cognitive models of memory, in particular constructive memory [10], which has the following distinguishing characteristics:

- memory is a reasoning process
- the index need not be explicit, it can be constructed from the query
- the index is changed by its use
- the content is changed by its use
- the memory structure is changed by its use
- memories can be constructed to fulfil the need to have a memory
- memories are a function of the interactions occurring at the time and place of the need to have a memory.

As can be seen constructive memory takes a fundamentally different view of memory than computational memory. Such memories are intimately connected to both the previous memories, called "experiences", the current need for a memory [14] and the current view of the world at the time and place of the need for that memory. Simple examples of some of these characteristics include the following [16].

## **5.1 Memory index changed by its use**

If the same query is made multiple times the response to the query should become faster, irrespective of whether it is a constructed index or not; this is a very simple example of how an index is changed by its use. A more profound and useful example of this phenomenon is when a memory is used to construct another, later, memory and a new index is created that connects these two memories such that when either is used again the other is associated with it.

## **5.2 Memory index can be constructed from the query, it need not be explicit**

Take the query: find an object with symmetry. There is no need for there to be an index “symmetry” in the memory. The system can use its experiences about symmetry to determine whether it can construct symmetry in objects in the memory. This concept allows for the querying of a memory system with queries it was not designed to answer at the time the original memories were laid down. This is significant in designing as there is evidence that designers change the trajectory of their designs during the process of designing and introduce new intentions based on what they “see” in their partial designs, intentions that were not listed at the outset of process. There are fundamental issues here that are not addressed by fixed index systems. For a novel query to be able to be answered it first needs to be interpreted by the memory system using the experiences it has that might bear on the query.

## **5.3 Memory content changed by its use**

A trivial example is in the case above about symmetry, the index can now become part of the memory. A less trivial example would be in the case where an experience is used in constructing a new memory. The experience is changed by having a link to the newly constructed memory. The experience is no longer the same experience it was before it was used to construct the new memory. That experience can no longer be recalled without its role in the construction of a new memory being part of it.

## **5.4 Memory structure changed by its use**

In the example above, not only does the content change but also the structure of the memory. The link between the experience and the new memory changes the structure of the memory system itself such that the way these memories can be used is changed. Later experiences can change what was experienced before in the sense that the associations between earlier experiences are changed by later experiences. The effect of this is that all experiences are potentially connected. Further, the meaning of an experience is a function of the situation within which it is used. The content of a constructive memory system is non-monotonic.

### **5.5 Memories can be constructed to fulfil the need to have a memory**

Take the case where a finite element analysis is carried out and both the cost of processing and the result are passed on to the design team leader. The team leader may query whether the analysis was cost effective. There is no memory of this but they can construct a memory in response. Later, if asked whether the analysis was cost effective they can respond directly through a recall-like process. If that query had not been asked earlier there would be no memory to respond to the later query.

### **5.6 Memories are a function of the interactions occurring at the time and place of the need to have a memory**

Take the example of the cost effectiveness query above. If at the same time as that query was being made another member of the design team states that his experience with the analysis group is that they always overestimate their costs, this changes the design team leader's construction of the memory to take account of some discounting of the cost provided.

## **6 Discussion**

The grand challenge of producing computational constructs that match our notions of designerly activity still remains. Can we build systems that are capable of answering such questions about designerly behavior as: How is it possible for a designer to commence a design without all the necessary information being available? How is it possible for a designer to continue designing when all the necessary information is not available? How is it that designers are able to produce novel solutions to what appear to be minor perturbations of existing design requirements? How is it possible for a designer to produce a different design when later presented with the same requirements?

It is claimed that situated design computing contains the seeds for the development of computational constructs that can be used to produce systems that match our notions of designerly activity. The two ideas of situatedness and constructive memory provide the basis for the development and use of experience in alternate situations. This is not a form of case-based reasoning or of default reasoning. It is a novel approach based on foundational concepts from cognition. This foundational behavior becomes increasingly designerly when we incorporate a design ontology such as the Function-Behavior-Structure ontology [13] into the memory system to provide it with the design domain knowledge structure.

Implementations of constructive memory systems require novel computational constructs that match the notion of construction. This involves the idea of "pull" which is similar to a goal-driven process except that it also changes the incoming data to match the perceptual expectations that support what is being constructed.

This is akin to making the interpretation match the expectation as opposed to checking whether the expectation can be met by the available data. In order to produce designerly behaviour we need to influence the development of computational systems rather than simply attempt to utilise what is already available.

## Acknowledgements

This research is supported by a grant from the Australian Research Council, grant number DP0559885 – Situated Design Computing. The research was carried out at the Key Centre of Design Computing and Cognition, University of Sydney.

## References

- [1] L. Suchman, "Plans and Situated Actions: The Problem of Human-Machine Communication", Cambridge University Press, 1987.
- [2] W. Clancey, "Situated Cognition", Cambridge University Press, 1997.
- [3] W. Clancey, "Conceptual Coordination", Cambridge University Press, 1999.
- [4] J. S. Gero, "Conceptual designing as a sequence of situated acts" in I. Smith (ed.), *Artificial Intelligence in Structural Engineering*, Springer, Berlin, pp. 165-177.
- [5] J. S. Gero, "Towards a model of designing which includes its situatedness" in H. Grabowski, S. Rude and G. Green (eds), *Universal Design Theory*, Shaker Verlag, Aachen, pp. 47-56.
- [6] P.E. Agre, "Computation and Human Experience", Cambridge University Press, Cambridge, UK, 1997.
- [7] G. Smith, G and J.S. Gero, "Interaction and experience: Situated agents and sketching", in J.S. Gero and F. Brazier (eds), *Agents in Design 2002*, Key Centre of Design Computing and Cognition, University of Sydney, Australia, pp. 115-132, 2002.
- [8] J. Zhang, "The nature of external representations in problem-solving", *Cognitive Science*, 21(2), 179-217, 1997.
- [9] J. Dewey, "The reflex arc concept in psychology", *Psychological Review*, 3, 357-370, 1896 reprinted in 1981.
- [10] J.S. Gero, "Constructive memory in design thinking", in G. Goldschmidt and W. Porter (eds), *Design Thinking Research Symposium: Design Representation*, MIT, Cambridge, pp. 1.29-35, 1999.
- [11] I. Rosenfield, "The Invention of Memory", Basic Books, New York, 1988.
- [12] E. von Glasersfeld, "Radical Constructivism: A Way of Knowing and Learning", The Falmer Press, 1995.
- [13] J.S. Gero, "Design prototypes: a knowledge representation schema for design", *AI Magazine*, 11(4), 26-36, 1990.
- [14] F.C. Bartlett, "Remembering: A Study in Experimental and Social Psychology", Cambridge University Press, Cambridge, 1932/1977.

- [15] J.S. Gero and U. Kannengiesser, "The situated Function-Behaviour-Structure framework", *Design Studies*, 25(4), 373-391, 2004.
- [16] J.S. Gero, "Understanding situated design computing: Newton, Mach, Einstein and quantum mechanics", in I. Smith (ed.), *Intelligent Computing in Engineering and Architecture*, Springer, Berlin, pp. 285-297, 2006.