COMPUTER-AIDED DESIGN TOOLS THAT ADAPT

WEI PENG
CSIRO, Australia

and

JOHN S GERO
George Mason University, USA

Abstract. This paper describes an approach that enables a computer-aided design tool to learn conceptual knowledge as it is being used, and as a consequence adapts its behaviours to the changing environment. This allows the tool to improve the effectiveness of designers in their design tasks over time. Design experiments evaluate the effectiveness of this prototype system in recognizing optimization problems in heterogeneous design scenarios.

1. Introduction

The development of computer-aided design tools has moved from representation to knowledge encoding and support in knowledge-based systems. A large number of design knowledge systems have been prototyped or commercialized, such as OPTIMA (Balachandran 1988) and KNODES (Rutherford and Maver 1994). Designing is intrinsically dynamic and interactive, in the sense that designers reflect on their actions (Schön 1983) and often change the course of the developing design (Gero 1998). Many CAD researchers turned to building systems that can automatically learn to cope with this ill-structured problem. Machine learning techniques have been widely adopted in knowledge-based systems to provide knowledge acquisition, modification and generalization, for example ECOBWEB (Reich and Fenves 1991) and BRIDGER (Reich 1993). These systems treat knowledge as universally applicable context-free generalizations and descriptions (Reffat and Gero 2000), so that they can be reused in different circumstances. It is argued that there are disadvantages of a black-box, context-free learning machine (Lieberman and Selker 2000).

A Dong, A Vande Moere & JS Gero (eds), CAADFutures'07, 417-430.

A fundamental question is how to enhance design effectiveness using computer-aided tools. The effectiveness of a design process is often associated with the term "efficacy" when a design tool is applied in a design activity. The efficacy of the tool usually refers to the ability to produce a desired amount of a desired effect. To enhance the efficacy of a CAD tool, we need a mechanism to bring changes in the system that are adaptive, in the sense that these changes enable the system to tackle the same task or tasks drawn from the same population more successfully the next time (Simon 1983). A design tool that adapts based on its experience of its use is claimed to be effective (Gero 2003). This paper describes an approach that enables a computer-aided design tool to be built on an adaptive paradigm, so that a design tool can learn conceptual knowledge as it is being used, and as a consequence adapts its behaviours to the changing environment. We present a computational model that is founded on notions of situatedness from cognitive science and computational agency.

2. Situatedness and Adaptation

The concept of "situatedness" has been developed in different areas resulting in diverse terms, such as "situated action" (Suchman 1987) and "situated cognition" (Clancey 1997). Situatedness involves both the context and the observer's experiences and the interactions between them. It is inseparable from interactions in which knowledge is dynamically constructed. Situated cognition copes with the way humans construct their internal worlds via its interaction with the external world (Gero 2003). The notion of adaptation originates from the ability of a biological system to accommodate incremental changes and to react to unexpected events in its environment. Adaptation can be viewed as the system's capability of modifying its behaviours to its context and improving its performance over time (Boer and Canamero 1999). The adaptive behaviour results from the interaction between an agent and its environment (Beer 1997). In this paper, we present a situated agent-based design tool, which consists of an existing design tool, a situated agent and interactions between the agent and its environment. A situated agent is wrapped around the design tool, learns from and adapts to its interactions with the design environment. Adaptation enables the design tool to cope with situatedness in a dynamic design process (Gero and Peng 2004). The concepts about interactions are constructed and grounded into the agent's experiences. These experiences bias the agent's later memory construction when a similar situation is encountered. The constructive memory model embodies a mechanism whereby an agent learns new concepts.

2.1. SITUATED AGENTS

A situated agent is a software agent built on the notion of "situatedness". Adaptivity, the agent's capability to learn and improve with experience (Bradshaw 1996), is a salient feature of a situated agent. A constructive memory model (Gero 1999) serves as an operational utility that implements the idea of "situatedness" into agent architecture. Adaptive behaviours, in terms of reflexive, reactive and reflective behaviour (Maher and Gero 2002), can result from the multi-level processing and constraints imposed by a situated agent architecture. Experience is a general notion that comprises the knowledge or skill of some thing gained through direct involvements or activities. This paper represents experience as structures. They can be classified into three categories:

1. The sensory experience holds discrete symbolic labels for discerning sense-data. They are the built-in features for sensors. Each sensor captures a particular type of information. Once an environment stimulus is detected, the agent attaches an initial meaning to it, based on its sensory experience;
2. The perceptual experience captures historical representations of perceptual categories and their interrelationships, including entities, properties and entity–property relationships with degrees of beliefs;
3. The conceptual experience comprises the grounded invariants over the lower level perceptual experience. The conceptual experience explicitly states the regularities over the past observations of perceptual instances.

Situated agents can sense and put forward changes to the environment via sensors and effectors. Sensors gather environmental changes into data structures called sense-data. Sensation (S) is the process that transfers sense-data into multi-modal sensory experiences. This is through "push" and "pull" processes. A push process is a data-driven process in which changes from the external world trigger changes in the agent's internal world, for example, the agent's experience. A pull process is an expectation-driven process in which the agent updates the internal world according to the expectation-biased external changes (Gero and Fujii 2000; Gero and Kannengiesser 2006). The push and pull processes can occur at different levels of processing, for example, sensation, perception and conception. The pushed sense-data are also called exogenous sense-data (S_e). They are triggered by external environmental changes, that is, actions performed by designers in using the design tool. The pulled sense-data are intentionally collected during the agent's expectation-driven process. Sensory data (S_{e+a}) consist of two types of variables: the exogenous sense-data (S_e) and the autogenous sensory experience (S_a). S_a is created from matching the agent's

exogenous sense-data (S_e) with the agent's sensory level experience. Sensory experience (S_{e+a}) are a combination of the agent's exogenous sense-data (S_e) and the related autogenous information (S_a). For instance, sense-data S_e is captured by sensors as a sequence of unlabelled events:

- S_e (t) = {…… "a mouse click on a certain text field", key stroke of "x", "y"……}.

Based on the lowest level of sensory experience, which holds modality information, the agent creates an autogenous variable (S_a) with its initial label for the S_e:

- S_a (t) = {"Label for the clicked text field"}.

Thus, sensory experience S_{e+a} can be created as:

- S_{e+a} (t) = {…… ["Label for the clicked test field" | Key strokes "x", "y"]……}

Perception (P) generates percepts based on the agent's sensory experiences. Percepts are intermediate data structures that are generated from mapping sensory data into categories. Sensory experience S_{e+a} is categorized to create initial percept (P_i) which can be used to generate a memory cue. The initial percept can be structured as a triplet "Percept (Object, Property, Values of properties)". It is expressed as:

- P_i (t) = Object {Property for the clicked test field, value of that property "xy"}

The perceptual object can be used to cue a memory of the agent's experience. A cue refers to a stimulus that can be used to activate the agent's experience to obtain a memory of that experience. It is generated from matching percepts with the agent's perceptual experience. A cue is subsequently assigned with an activation value to trigger responses from the agent's experience. The cueing function is implemented using experience activation (I_a) and reactivation (I_r), in which a memory cue is applied to the experience structure to get a response.

Conception is the process of categorizing perceptual sequences and chunks in order to form proto-concepts. A concept is regarded as a result of an interaction process in which meanings are attached to environmental stimuli. In order to illustrate a concept formation process, we use the term "proto-concept" to illustrate the intermediate state of a concept. A proto-concept is a knowledge structure that depicts the agent's interpretations and anticipations about its external and internal environment at a particular time. Conception consists of three basic functions: conceptual labeling (C_1), constructive learning (C_2) and induction (C_3). Conceptual labeling creates proto-concepts based on experiential responses to an environment cue. This

includes deriving anticipations from these responses and identifying the target. Constructive learning allows the agent to accumulate lower level experiences. Induction can generalize abstractions from the lower level experience and is responsible for generating conceptual knowledge structures.

The hypothesizing process (H) generates a hypothesis from current learned proto-concepts. It is where reinterpretation takes place in allowing the agent to learn in a "trial and error" manner. A situated agent reinterprets its environment using hypotheses which are explanations that are deduced from its domain knowledge (usually conceptual). An agent needs to refocus on or construct a new proto-concept based on hypotheses. Validation (V_d) is the process in which the agent verifies its proto-concepts and hypotheses. It pulls information from the environment to observe whether the environment is changing as expected. A valid concept or experience will be grounded into experiences by incorporation or reinforcement.

The grounding process refers to the experiential grounding (Liew 2004). This reinforces the valid concepts or activated experience via changing the structures of the experience so that the likelihood of the grounded experience being activated in similar circumstances is increased. This is implemented by a grounding via weight adaptation process (W_a), which adjusts the weights of each excitatory connection of the valid concept of a Constructive Interactive Activation and Competition (CIAC) neural network (Peng and Gero 2006), which is an extension of IAC neural network (McClelland 1981), so that those nodes that fire together become more strongly connected.

2.2. ADAPTIVE BEHAVIOURS AND LEARNING MECHANISMS

The agent's reflexive behaviour occurs at a macroscopic level when the experiential response to current sensed data is sufficiently strong to reach a reflexive threshold. A sensory experience can affect action directly. In this circumstance, the agent reflexes to environment stimuli based solely on its experience without activation. In its reactive mode, an agent applies its perceptual experience to respond to an environment stimulus in a self-organized way. The perceptual experience, in terms of a habitual sequence of actions, is manifested as an initial concept. The agent reflects on its actions by drawing new sense-data from a lower level and hypothesizing a new concept. A situated agent reflexes, reacts or reflects corresponding to concepts constructed from its constructive memory model. "Situatedness" emphasizes the role of social relations and interactions in learning. An agent that is designed to be situated at a conceptual level can be implemented using various machine learners. The situated learning can be studied on the following two levels: 1. At a meta-level, learning refers to the concept

formation process arising from a constructive memory model for a situated agent; 2. At a base-level, various concept formation composites can be modeled via various machine learners, for example, connectionist neural networks, inductive and analytical machine learning algorithms. The learning process is the process wherein the agent constructs new concepts, such that the agent's experiences (as structure) are reinterpreted, re-structured and constructed in the current context.

3. Situated Agent-based Design Optimization Tool

This research is presented within the design optimization domain. Many design optimization tools focus on gathering a variety of mathematical programming algorithms and providing the means for the user to access them to solve design problems.[1] Choosing a suitable optimizer becomes a bottleneck in a design optimization process. The recognition of appropriate optimization models is fundamental to design decision problems (Radford and Gero 1988). Some of the knowledge required for the recognition of an optimization problem can be expressed in terms of semantic relationships between design elements. An example of such knowledge is illustrated in Table 1. The application of this research to design optimization focuses on learning and adapting the knowledge of applying various optimization algorithms in different design contexts. For example, a designer working on optimizing a hospital layout may find that a certain optimizer is more efficient in solving the problem applied. As the same or other designers tackle a similar design problem, the same tool draws on its experience to construct memories of a design situation and anticipates the tool's potential use. It can offer help to designers in their interactions in designing even before they call for it.

TABLE 1. Knowledge in choosing an optimizer (after Radford and Gero (1988))

if	all the variables are of continuous type
and	all the constraints are linear
and	the objective function is linear
then	conclude that the model is linear programming
and	execute linear programming algorithm

We further discuss a scenario that depicts potential impacts of such a situated agent-based design tool in a design optimization process. Under normal circumstances, a designer uses a design optimization tool to define and solve a problem. No matter how many times he or she applies the same tool to address similar design problems the tool remains unchanged from its

1 http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/.

use. The designer has to repeat each step each time. We suggest that there are potential benefits if design knowledge can be learned and become available for use without repeating the often demanding design optimization process (Radford and Gero 1988).

A design trajectory consists of a sequence of actions performed by a designer. It represents the procedure via which a design problem has been solved. An assumption here is that the system has already gained certain experiences in design optimization. This assumption is realistic from what we have seen in previous sections. For example, the knowledge of the optimality for the unconstrained quadratic programming problem may contain associative rules like:

- Hessian matrix (positive-definite) → Local-min achieved;
- Hessian matrix (indefinite) → Saddlepoint achieved.

When a designer is keying in a quadratic objective function, the system forms a concept derived from the constructed memory (assuming there exists a similar design optimization instance). According to the problem it recognizes, the system can present the anticipated steps to remind the designer. These include suggestions like (also shown in Figure 1 as "1", "2" and "3" of the concept formed from the system's reactive behaviour):

1. "may be a quadratic programming problem";
2. "may look at Hessian function and second-order of Hessian function to decide its type";
3. "may be a local minimum because Hessian matrix is positive-definite from the system's memory of a similar design instance, use medium-scale quadratic optimizer";

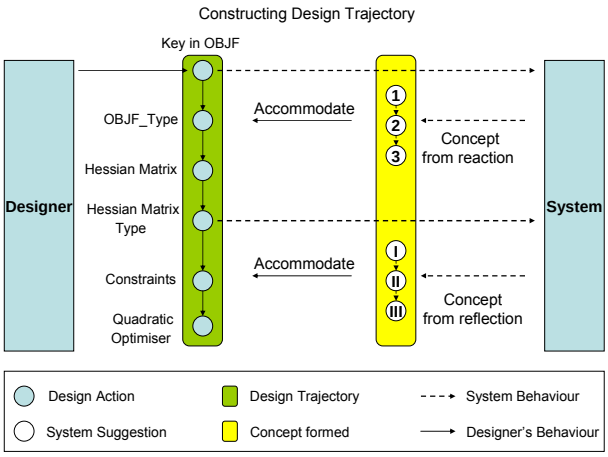

Figure 1. A scenario of constructing a design trajectory in interactions

This concept may guide the designer to focus his or her attention on the contextual information that is drawn from the system's experience on similar problems. This may reduce repetitions in solving a design problem. The system can then observe the designer's actions in deciding its subsequent moves. If the designer works out the Hessian matrix to be indefinite, the system can draw on the knowledge of the optimality to deduce a possible explanation (also shown in Figure 1 as "I", "II" and "III" of the concept formed from the system's reflective behaviour):

I. "may still be a quadratic programming problem";
II. "may be a saddle point because Hessian matrix is indefinite from the system's memory of a similar design instance, don't forget constraints if there are";
III. "may use large-scale quadratic optimizer".

In this way, the concept formed by the system can be infused into the designer's actions. A design trajectory can be constructed and modified in the interactions. As illustrated in Figure 1, the concepts formed from the system's reactive and reflective behaviour are accommodated into a designer's design trajectory. The tool that maintains such a predictive model based on valid anticipations may improve the efficacy of a design process through introducing the agent's experience in developing the design outcome. The efficacy of such a design tool can be measured through its correctness in recognizing a design optimization problem.

4. Design Optimization Experiments

The implemented prototype system is applied to assist the use of a design optimization tool (the Matlab Optimization Toolbox). Matlab Optimization Toolbox is a collection of functions that extend the capability of the MATLAB numeric computing environment. The toolbox includes routines for a variety of optimization classes, including unconstrained and constrained nonlinear minimization, quadratic and linear programming, and nonlinear optimization. It has been widely used by engineers in various domains. A situated agent learns knowledge from how Matlab is utilized by a designer in solving various optimization problems and uses the learned concepts to affect the tool's future use. This section presents a number of experiments that have been carried out on the implemented prototype system. The basic assumption for the experiments is that a user has already worked out the objective function and constraints; he or she uses a design tool to solve that problem. The purpose of the experiments is to evaluate the situated agent-based design tool through:

- examining whether the system can learn new concepts from interactions;
- investigating whether the implemented model can develop adaptive behaviours in different circumstances based on the knowledge structures it learned; and
- studying the characteristics of the agent's behaviours in various circumstances and evaluating the efficacy of the implemented prototype system.

4.1. EXPERIMENT RESULTS

This test (Test 1) focuses on observing and analyzing the agent's behaviours in heterogeneous design optimization scenarios. A sequence of 15 design scenarios is created and adopted. Each scenario represents a design task which is further composed of a number of design actions. For example, a typical design optimization task consists of a number of actions:

- defining objective function and identifying objective function type;
- defining design variables, variable types, design constraints and constraint types;
- typing in gradients of objective function and constraints, defining matrices, such as Hessian matrix and its type;
- selecting optimizers, submitting design problem or editing design problem; submitting feedback on agent's outputs.

The sequence of 15 tasks is represented as {L, Q, Q, L, NL, Q, NL, L, L, NL, Q, Q, L, L, L}, in which "Q", "L" and "NL" represent quadratic, linear and nonlinear design optimization problems respectively. The initial experience of the agent holds one instance of a design optimization scenario solved by a quadratic programming optimizer. The symbols in Table 2 represent these behaviours. According to data obtained from this test, we can further cluster the system's learning behaviour into three stages: Stages I, II and III. We use behaviour rate (B_r) to measure distributions of various behaviours in each stage. The behaviour rate (B_r) for each stage is defined as:

$$B_r = \frac{Numbers\ \ of\ \ a\ \ particular\ \ \ behaviour}{Total\ \ numbers\ \ of\ \ behaviours\ \ \ in\ the\ \ stage}$$

TABLE 2. Symbols that represent various behaviours

SYMBOLS	BEHAVIOURS (B_E)	DESCRIPTIONS
C_1	Conception process 1 – conceptual labelling	Focusing on the target concept from the activated experience
C_2	Conception process 2 –	Creating perceptual experience from memory

	conception via constructive learning	construction (constructive learning)
C_3	Conception process 3 – conception via inductive learning	Creating conceptual experience from generalization (inductive learning)
H	Hypothesizing	Deducing proto-concepts from hypotheses
I_a	CIAC neural network activation	Activating the perceptual experience structure (CIAC) to get response
I_r	CIAC neural network re-activation	Re-activating the perceptual experience structure (CIAC) to get response
P	Perception	Low-level behaviour in creating percepts and memory cue
R_{ex}	Reflexive experience response	Returning experience that reaches reflexive threshold (no reasoning and activation required)
S	Sensation	Low-level behaviour in creating sensory data
V_d	Validation	Comparing anticipation with environment changes
W_a	Weight adaptation	Reinforcing the experience when it is useful

The B_r of a particular behaviour represents the frequency of this behaviour in the learning stage in which it occurs. The results of various B_r for the three stages are presented in Figures 2-4. Stage I consists of Tasks 1 to 5. No high-level experience or processes (C_3, H) are involved in this stage. The system reacts and learns via C_2 (constructive learning), as depicted in Figure 2.

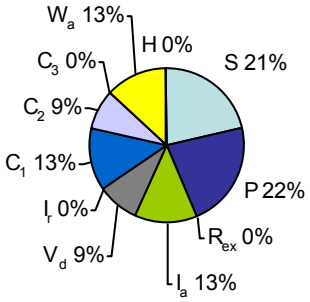

Figure 2. Agent behaviour in learning Stage I

In Stage II (tasks 6 to 12), high-level processes, such as reactivation (I_r), inductive learning (C_3) and hypothesizing (H) become dominant and the system is concentrated on reflection, Figure 3. In Stage III (tasks 13 to 15), the experience for a certain type of design optimization problem becomes highly grounded and the system commences its reflexive behaviour, as illustrated in Figure 4.

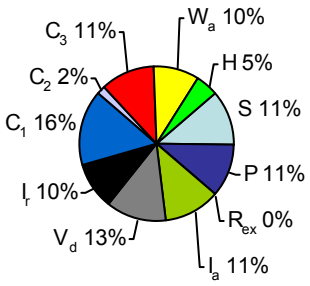

Figure 3. Agent behaviour in learning Stage II

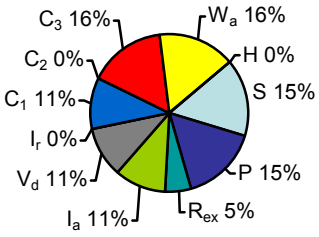

Figure 4. Agent behaviour in learning Stage III

A comparative study of these learning stages shows a higher percentage of C_2 (constructive learning) in Stage I (9%, compared with 2% for Stage II and 0% for Stage III). This means that the system is in the initial stage of learning – constructing new memories. There are no high-level behaviours (I_r, H, C_3) and much higher percentages of sensation (S) and perception (P) in the initial stage of learning (Stage I). The system's behaviours are more low-level oriented at this point, due to the lack of resources in generalization. With conceptual knowledge being formed at the beginning of Stage II, the system manifests a reflective behaviour in which it revisits its experience to reactivate and make hypotheses.

As illustrated in Figure 3, the agent's reflection-related behaviours, such as H and I_r contribute to 5% and 10% of its overall behaviours, compared to 0% in other stages. The salient feature for Stage III is that the system demonstrates a higher percentage of reflexive behaviour (5% against 0%) than those in the other two stages. Stages I, II and III are similar in reaction, validation and grounding related behaviours, such as I_a V_d and W_a, because the system has similar proportions of grounded reactive experience. This three-stage taxonomy can be explained by the internal structures created in

the experiment. Conceptual knowledge is learned at task 6, which is the grounded commonality over the incrementally gathered perceptual experience (from the CIAC neural network). This concept enables the system to create hypotheses and therefore contributes to the system's reflective behaviour at Stage II. At the end of task 14, the experience for the linear optimization problem is so strong that it is on the threshold of producing the reflexive behaviour in Stage III.

4.2. A COMPARISON TEST

In this test, we investigate the performance of three systems: a static system, a reactive system and a situated system, in learning to recognize design optimization problems. The design scenario of Test 1 is adopted. A static system can only use the predefined knowledge to predict a design task. A reactive system can use *a priori* knowledge to respond to an environmental cue. It can also learn via constructive learning, provided it encounters a new design problem. A situated system not only employs its existing experience to react, it also reflects using the hypotheses created based on the accumulated conceptual knowledge. The performance is defined as the correctness of the system's response to an environmental cue, which predicts an interaction situation, and hence assists the applied design task. We use prediction success rate (P_s) to measure the overall performance of a system in this test:

$$P_s = \frac{Number\ \ of\ \ correct\ \ prediction\ s}{Total\ \ numbers\ \ of\ \ prediction\ s\ in\ the\ test}$$

The prediction success rate corresponds to the percentage of correctly predicted examples over total test examples. Based on the results measured from this test, we can calculate prediction success rates for each system. A situated system produces a prediction success rate of *0.80*, followed by the rates of *0.67* for the reactive system and *0.33* for the static system respectively. We conjecture the reason for this is the ability of a situated system to generalize across observations and subsequently to deduce explanations for environmental changes. It is also noted that the agent uses the conceptual knowledge to hypothesize and reflect from Task 10, thus providing better performance from that point.

5. Conclusion

Experimental results show that the implemented system can learn new concepts through its use in interactions in design optimization. Another finding is that the agent can develop adaptive knowledge structures through constructing a memory, during which the agent coordinates the system's

experience and environmental context in a situated manner. The system exhibits adaptive behaviours to this end. Compared to a static system based on pre-defined knowledge and a reactive agent which learns by the constructive learning, this situated agent-based design interaction tool performs better. In summary, the approach plays potential roles in enhancing design effectiveness through introducing mechanisms that enable a computer-aided design tool to adapt based on its experience of its use in a dynamic design process. The framework developed here may also lay foundations for future research into adaptive and personalized design tools.

Acknowledgements

This work was supported by a CRC-CI Scholarship, a University of Sydney Sesqui R&D grant and partly carried out at the Tasmanian ICT Centre which is jointly funded by the Australian Government through the Intelligent Island Program and the CSIRO. The Intelligent Island Program is administered by the Tasmanian Department of Economic Department.

References

Balachandran, MB: 1988, *A Model for Knowledge-Based Design Optimization*, PhD Thesis, University of Sydney, Sydney.

Beer, RD: 1997, The dynamics of adaptive behaviour: A research program, *Robotics and Autonomous Systems* **20**: 257-289.

Boer, B and Canamero, D: 1999, Situated learning in autonomous agents, *in* J Joan Bliss, R Saljo and P Light (eds), *Learning Sites: Social and Technological Resources for Learning*, Pergamon, Amsterdam, pp. 236-248.

Bradshaw, J (ed.): 1996, *Software Agents,* MIT Press, Cambridge.

Clancey, W: 1997, *Situated Cognition: On Human Knowledge and Computer Representations,* Cambridge University Press, Cambridge.

Gero, JS: 1998, Conceptual designing as a sequence of situated acts, *in* I Smith (eds), *Artificial Intelligence in Structural Engineering*, Springer, Berlin, pp. 165-177.

Gero, JS: 1999, Constructive memory in design thinking, *Design Thinking Research Symposium: Design Representation,* MIT, Cambridge, pp. 29-35.

Gero, JS: 2003, Design tools as situated agents that adapt to their use, *in* W Dokonal and U Hirschberg (eds), *eCAADe21,* eCAADe, Graz University of Technology, pp. 177-180.

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: 2006, A framework for situated design optimization, *Design & Decision Support Systems 2006,* Springer-Verlag, Berlin, in press,

Gero, JS and Peng, W: 2004, A situated agent-based design assistant, *CAADRIA 2004,* Yonsei University Press, Korea, pp. 145-157.

Lieberman, H: 2001, Introduction, *in* H Lieberman (eds), *Your Wish is My Command: Programming by Example*, Morgan Kaufmann, San Francisco, pp. 1-7.

Liew, P-S: 2004, *A Constructive Memory System for Situated Design Agents*, University of Sydney, Sydney.

Maes, P: 1994, Agents that reduce work and information overload, *Communications of the ACM* **37**: 31-40.

Maher, ML and Gero, JS: 2002, Agent models of 3D virtual worlds, *ACADIA 2002: Thresholds,* California State Polytechnic University, Pomona, California State Polytechnic University, Pomona, pp. 127-138.

McClelland, JL: 1981, Retrieving general and specific information from stored knowledge of specifics, *Proceedings of the Third Annual Meeting of the Cognitive Science Society,* Erlbaum, Hillsdale, NJ, pp. 170-172.

Peng, W and Gero, J: 2006, Using a constructive interactive activation and competition neural network to construct a situated agent's experience, *PRICAI 2006: Trends in Artificial Intelligence, Ninth Pacific Rim International Conference on Artificial Intelligence,* Springer, Guilin, pp. 21-30.

Radford, AD and Gero, JS: 1988, *Design by Optimization in Architecture and Building,* Van Nostrand Reinhold, New York.

Reffat, R and Gero, JS: 2000, Computational situated learning in design, *in* JS Gero (eds), *Artificial Intelligence in Design'00*, Kluwer Academic Publishers, Dordrecht, pp. 589-610.

Reich, Y: 1993, The development of BRIDGER: A methodological study of research in the use of machine learning in design, *Artificial Intelligence in Engineering* **8**(3): 165-181.

Reich, Y and Fenves, S: 1991, The formation and use of abstract concepts in design, *in* D Fisher, M Pazzani and P Langley (eds), *Concept Formation: Knowledge and Experience in Unsupervised Learning*, Morgan Kaufmann, San Mateo, CA, pp. 323-353.

Rutherford, JH and Maver, TW: 1994, Knowledge-based design support, *in* G Carrara and YE Kalay (eds), *Knowledge-Based Computer-Aided Architectural Design*, Elsevier Science, Amsterdam, The Netherlands, pp. 243-267.

Schön, D: 1983, *The Reflective Practitioner: How Professionals Think in Action,* Basic Books, London.

Simon, HA: 1983, Why should machine learn, *Machine Learning: An Artificial Intelligence Approach,* Springer-Verlag, Berlin, pp. 25-37.

Suchman, LA: 1987, *Plans and Situated Actions: The problem of human-machine communication,* Cambridge University Press, Cambridge.