INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

LEARNING FIRST-PERSON KNOWLEDGE THROUGH INTERACTIONS: TOWARDS EFFECTIVE DESIGN TOOLS

Wei Peng[1], John S. Gero[2]

[1]Tasmanian ICT Centre, CSIRO ICT Centre, Australia
[2]Krasnow Institute for Advanced Study and Volgenau School of Information Technology and Engineering, George Mason University, USA

ABSTRACT

This paper introduces an approach that enables a design tool to learn first-person knowledge through interactions. A design tool of this kind embodies learning and adaptive behaviours. The contextual knowledge in using the tool to solve a particular design problem can be captured, made available and adapted to a designer's decision-making when he or she is confronted with a similar or a new problem at a later time. An effect of this is an improvement in design efficacy, in that a design task can be recognized from interactions and experiences for similar tasks and support can be provided to aid decision-making. The implemented prototype system is applied to assist the use of an optimization tool (the Matlab Optimization Toolbox) in design. The system learns knowledge from how Matlab is utilized in solving various optimization problems and uses the learned concepts to affect the tool's future use. Experiments evaluate the effectiveness of this prototype system in recognizing optimization problems in various design optimization scenarios.

Keywords: First-Person Knowledge, Design Tool, Situated Agent, Interaction, Design Optimization

1 INTRODUCTION

Computer-aided design tools (CAD) were first introduced to assist designers to evaluate the "goodness" of their creations [1]. They now extend their functionality to include three-dimensional modelling, computer simulation, analysis, and integration between applications. The effectiveness of a design tool, which is often associated with the term "efficacy", refers to the ability to produce a desired amount of intended effects. Contemporary design tools continue to be built based on a paradigm that the tool is unchanged by its use [2], [3]. These design tools keep repeating themselves, irrespective of their interactions with the design environment. The functions are hard-coded during the development stage as third-person knowledge or engineering science knowledge. Their ability to assist a designer in a dynamic design process is limited due to:

1. the fixedness this third-person knowledge;
2. the inability to include designer's experience and;
3. the inability to take account of interactions in the design process, in which designers interact with their environments in developing designs.

Whilst most research and practice are focused on addressing the first two issues, for example, in knowledge-based design systems, design education and training, we present an approach to improve the effectiveness of a design tool by addressing the third point. First-person knowledge, i.e., knowledge that a knower would express by a first-person sentence, has been studied in the philosophical area of phenomenology [4]. Exploring the source of first-person knowledge entails our concerns of knowing how we experience everyday. First-person knowledge from this perspective enables us to have goals, to evaluate them and to change them; it therefore underwrites our rational agency [5]. There are a number of hypotheses which are aimed to explain the possible source of first-person knowledge, for example introspection and cognitive transformation from *a priori* knowledge [4]. We are concerned with developing a computational model that learns first-person knowledge from a constructivist point of view. Interaction between a software agent and its environment serves as a source via which first-person knowledge can be constructed.

This paper introduces an approach that enables a situated agent-based design tool to learn first-person knowledge through interactions. A design tool of this kind embodies learning and adaptive behaviours. It is claimed that interaction as one of the fundamental characteristic of a design process can be enhanced by taking account of the first-person knowledge that is developed based on first-person interaction with the environment. The contextual knowledge in using the tool to solve a particular design problem can be captured, made available and adapted to a designer's decision-making when he or she is confronted with a similar or a new problem at a later time. This approach draws concepts from situated cognition [6] to develop a computation model of a design interaction tool [7], which learns by its use. A situated agent wraps around a design tool and constructs concepts from the interactions between the agent, the design problem and the use of the tool [8]. A situated system uses a constructive memory model [9] to create anticipations which can go beyond contextual information and guide design interactions. We explore the effect of a situated agent in relation to design efficacy, in that a design task can be recognized from interactions and experiences for similar tasks and support can be provided to aid decision-making. This research is presented within a design optimization domain.

2 DESIGN OPTIMIZATION APPLICATION DOMAIN

Design optimization is concerned with identifying optimal design solutions which meet design objectives while conforming to design constraints. The formal statement of the optimization problem can be denoted as [10]:

$$\textit{Minimize } f(x) \tag{1}$$

Subject to:

$$h(x) = 0 \tag{2}$$

$$g(x) \leq 0 \tag{3}$$

$$x \in \chi \subseteq R^n \tag{4}$$

where in expression (1), the scalar objective function $f(x)$ is the criterion among different alternatives. The vector-valued functions $h = (h_1, h_2, \ldots, h_{m1})^T$ and $g = (g_1, g_2, \ldots, g_{m2})^T$ are the functional constraints which are described in equations (2) and (3) as $h(x)$ and $g(x)$. As shown in expression (4), x is the n-dimensional vector of design variables belonging to a subset χ of the n-dimensional real space R^n.

Many research works in design optimization focus on providing new algorithms to improve the efficiency for the process of searching for optimal designs [11], [12]. As a consequence, a large variety of new optimization algorithms have been developed and are commercially available. Many design optimization tools target gathering a variety of mathematical programming algorithms and providing the means for the user to access them to solve design problems.[1] For example, Matlab Optimization Toolbox 3.0[2] includes a variety of functions for linear programming, quadratic programming, nonlinear optimization and nonlinear least squares, etc. Designers rely on the experience that they have built up through years of practice and use of these optimization tools to optimize a design. The outcome of this design process is constrained by their design knowledge. Recent research [13] identifies a number of issues that have not been well-addressed in a design optimization process:

1. lack of transfer of earlier results as the design changes;
2. lack of domain knowledge in computational tools;
3. lack of task knowledge in computational tools;
4. lack of feedback into process strategies in the tool.

Knowledge-based design systems and machine learning techniques emerged in a wide range of design areas (including design optimization) to assist the design decision-making process [14], [15], [16], [17]. These systems showed potential in providing knowledge-based support in design. However, these researches do not integrate "interaction" in the design model, which is viewed as a critical notion that provides the opportunity for change and adaptation – both in the internal knowledge of an

[1] http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/.
[2] http://www.mathworks.com/products/optimization/.

optimization system and in the design it is operating on [13]. Interaction in design optimization establishes the relationships between the designer, the problem formation, the tool and the result. Figure 1 introduces the concept of interactions as the key element of a design optimization process. A designer interacts with its environment which encompasses the design tool, the design problem and the design result in developing a design. Such a process consists of sequence of situated acts [18].

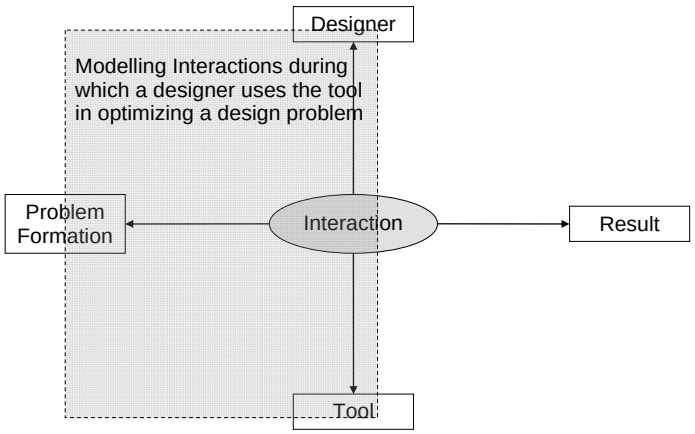

Figure 1. Interaction-based view of design optimization (adapted from Figure 3. [13])

Interactions provide a means for design optimization tools to flexibly construct task knowledge and domain knowledge that is adapted to the classes of optimization problems they are exposed to during those interactions [13]. Our aim is to construct a computational model that is able to learn first-person knowledge from using the tool and adapt the knowledge to the use of the tool while a designer is optimizing a design, and as a consequence offers assistance to the designer in his or her interactions in designing. We model the section of interactions during which a designer uses a tool to solve an optimization problem (shown as the grey square area enclosed by dashed lines in Figure 1). The implemented prototype system is applied to assist the use of an optimization tool (the Matlab Optimization Toolbox) in design. The system learns knowledge from how Matlab is utilized in solving various optimization problems and uses the learned concepts to help a designer in identifying various design optimization problems during the tool's later use. The identification of optimization problems is fundamental to the design optimization process [19]. Some of the knowledge required for recognition of the optimization problem can be expressed in terms of semantic relationships between design elements. An example of such knowledge is illustrated in Table 1. Design efficacy can be measured through the correctness in recognizing optimization problems in heterogeneous design scenarios.

Table 1. An example of knowledge required in recognition of an optimization problem (after [19])

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

3 SITUATEDNESS AND CONSTRUCTIVE MEMORY

This research is founded on two basic concepts: "situatedness" and "constructive memory". The concept of "situatedness" has its roots in works of empirical naturalism [20] and cognitive psychology [21]. It has been investigated in many different areas with diverse terms, such as "situated action" [22] and "situated cognition" [6]. Vygotsky contributed to the concept of "situatedness" through activity theory: defining that activities of the mind cannot be separated from overt behaviour, or from the social context in which they occur. Social and mental structures interpenetrate each other [23], [24]. Situatedness involves both the context and the observer's experiences and the interactions between

them. Situatedness is paraphrased as "where you are when you do what you do matters" [18]. It is inseparable from interactions in which knowledge is dynamically constructed.
Memory in computational systems often refers to a place that holds data and information called "memories". It is indexed so as to be queried more efficiently afterwards. However, we utilize a different theory of memory. In Clancey's review of Rosenfield's *The Invention of Memory*, he emphasized that memory is not a place where descriptions of what we have done or said before are stored, but is indistinguishable from our capability to make sense, to learn a new skill, to compose something new [25]. This is the essence of Bartlett's model of constructive memory [21]. The notion of a constructive memory reflects how a system adapts to its environment [26]. A constructive memory model [9] provides a conceptual framework for us, within which we may utilize the concept of "situatedness" in a software agent. "Memories are constructed initially from that experience in response to demands for a memory of that experience but the construction of the memory includes the situation pertaining at the time of the demand for the memory" [9]. Two operational characteristics of a constructive memory model are constructive learning and experiential grounding mechanisms [27]. Constructive learning is the means that an agent utilizes to develop its experience. It has an effect that brings changes in the structure of the memory system. This allows an agent to accommodate a new experience in an *a posteriori* manner. Experiential grounding is concerned with the provision of meanings to the experiences processed by an artificial agent [27]. It is similar to historical grounding [28], which considers the consequence of the utility of an experience in determining its meaning. According to Liew and Gero [27], the basic operations for a constructive memory model consist of:

- cueing[3]: the memory system is initially cued by a demand from the current situation;
- activation and selection: multiple experiences are activated, with only one being selected;
- memory construction: memory is constructed based on the selected experience; and
- incorporation: the constructed memory is incorporated into the system;

Situatedness and constructive memory entail the way in which first-person knowledge is constructed and grounded into experience.

4 GROUNDING EXPERIENTIAL ANTICIPATIONS IN SITUATED AGENTS

Symbolic grounding explores the means that the semantic interpretation of a symbol system can be made intrinsic to that system rather than relying on the meanings in the head of an external interpreter or observer [29]. An agent grounds its behaviour and representations in its interaction with the environment. The behaviour of the agent is intrinsically meaningful to itself [30]. Experiential grounding [27] had been proposed as a process that verifies the usefulness of a related experience in the current situation. It has the effect of increasing the likelihood of a previously cued memory being re-actived in the current time. Memory here is the computational construct of the cognitive concept of constructive memory [21]. In this paper, grounding of anticipation refers to the evaluation of whether a constructed anticipation correctly predicts environmental changes. We present how concepts can be formed through grounding anticipations in situated agents.

4.1 Situated Agency

A situated agent is a software agent that is founded on the notion of "situatedness". Situatedness in a software agent can be modeled as the interaction of three worlds [31], Figure 2. The *external world* denotes the world that consists of external events outside the agent. The *interpreted world* is the world constructed inside the agent. This internal world is composed of sensory experiences, percepts and proto-concepts. It is created through interpretation, where the intermediate percepts are transferred into or constructed as proto-concepts, which depict the initial meaning attached to the environment events as anticipations. The *expected world* is derived from the interpreted world. Through a process called *focusing*, the agent focuses on some aspects of the interpreted world, e.g. proto-concepts, and derives anticipations that predict future states of the external world. The process of *action* affects the external world based on anticipations constructed in the agent's self-organized *focusing* or *refocusing* processes. We hold that situatedness not only involves a recursive linkage between these three worlds, it is often accompanied by structural and behavioural adaptation in the interpreted worlds. An external change may cause the interpreted world to change. This can be achieved by activating or reactivating

[3] A cue refers to a stimulus that can be used to activate the agent's experience to obtain a memory of that experience.

the agent's experience. With reinterpretations, the agent can refocus or create a new anticipation to affect the external world. Based on the grounding of anticipations, the agent can develop concepts and adaptive behaviours over time.

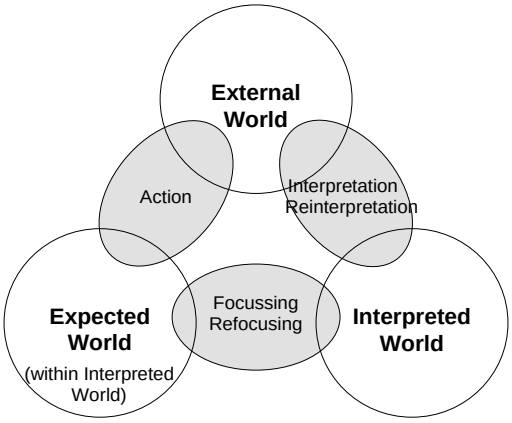

Figure 2. Situatedness as interaction of three worlds (after Fig. 2. [31])

A situated agent contains sensors, effectors, experience and a concept formation engine, which consists of a perceptor, a cue_Maker, a conceptor, a hypothesizer, a validator and related processes. Sensors gather events from the environment. In the following example these events include key strokes of objective functions, the users' selections of design optimization algorithms, etc. Sense-data takes the form of a sequence of actions and their initial descriptions. For instance, sense-data can be expressed as:

$$S(t) = \{ \ldots\ldots \text{“click on objective function text field”, key stroke of “x”, “(”, “1”, “)”, } \ldots\ldots \} \quad (5)$$

The perceptor processes sense-data and groups them into multimodal percepts, which are intermediate data structures illustrating environment states at a particular time. Percepts are structured as triplets:

$$P(t) = \{Object, Property, Values\ of\ properties\} \quad (6)$$

For example, a perceptual data P_1 can be described as {Objective Function Object, Objective_Function, “2x(1)+x(2)”}. The cue_Maker generates cues that can be used to activate the agent's experience. Anticipation generation is the process that is associated with this activation. Anticipation generation is the process of generating anticipation about potential environment changes. Anticipation is related to the agent's view about possible consequences of certain actions and affects its decision-making. When an unexpected condition is recognized, it needs to be reinterpreted [32]. Reinterpretation occurs in the hypothesizing process, in which focused concepts are selected for anticipation generation and the causalities of failures are located in order to modify the anticipations. The hypothesizer generates a hypothesis from current learned proto-concepts. It is where reinterpretation takes place to allow 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 its hypotheses.

The conceptor categorizes the agent's experience to form concepts. Conception is the process running in the conceptor. Conception consists of three basic functions: conceptual labelling (C_1), constructive learning (C_2) and induction (C_3). Conceptual labelling 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 generalise abstractions from the lower level experience and is responsible for generating conceptual knowledge structures.

The validator pulls information from the environment and examines whether the environmental changes are consistent with the agent's anticipations. An agent needs to validate its hypotheses in interactions. An effector is the unit via which the agent brings changes to environments through its actions.

4.2 Interpretations and Anticipations

Research results from cognitive development stress an inextricable link between contextual constraints and the acquisition of knowledge. The contemporary view treats cognition as typically situated in a social and physical context [33]. A major concern for this study is to build a link between context and the agent's situated concepts formed from interactions – first-person constructs. Learning concepts from contexts involves processes that construct the agent's interpretations and anticipations about a context, and in turn validates the constructed concepts based on their usefulness in predicting and affecting a context in time. These processes include sensation, perception, anticipation generation, conception, hypothesising, action and validation. Here we discuss three scenarios that show how a context can be interpreted (or re-interpreted) and constructed into a concept. The interpretation is concerned with associating an initial meaning with a context. Anticipation is the process of an agent making decisions based on predictions, and expectations about the future.[4] The anticipation refers to predicting future states of the environment based on the interpretation of the context.

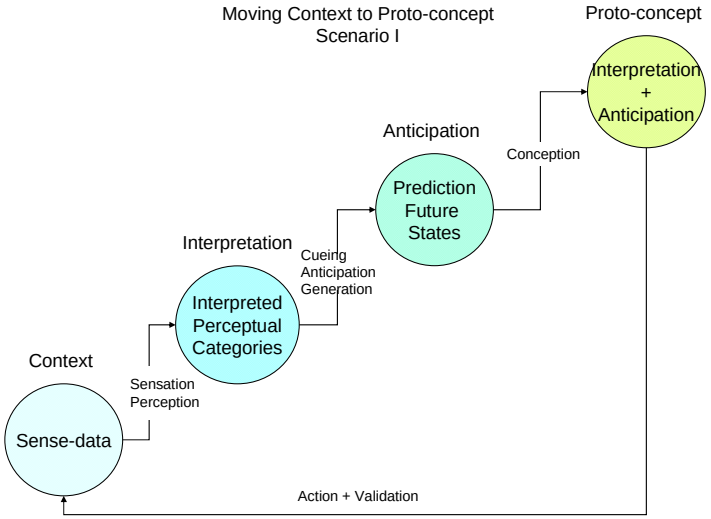

Figure 3. Scenario I, in which the agent moves a context into a proto-concept

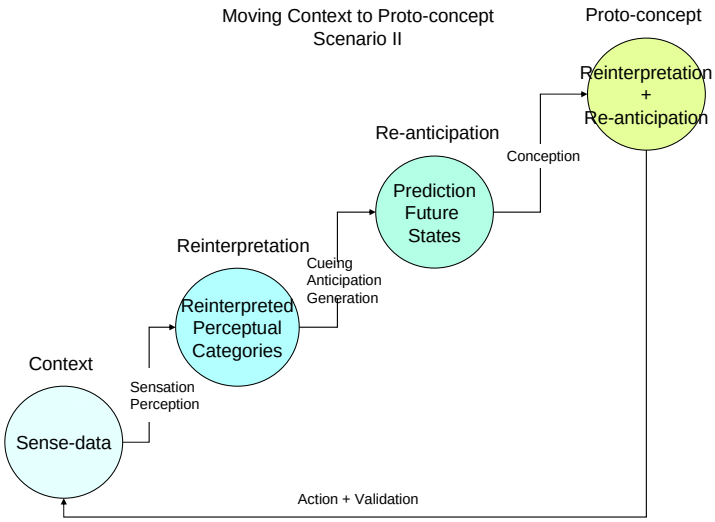

Figure 4. Moving a context into a proto-concept, Scenario II

Figure 3 presents a scenario in which a context in terms of sense-data is constructed into a proto-concept. Through sensation and perception processes, the contextual information is interpreted based on the agent's experience. With this initial transformation, the agent creates a mapping between the context from its external world and its internal world. It can therefore cue its memory to generate

[4] http://en.wikipedia.org/wiki/Anticipation.

anticipations for the interpreted perceptual category. The conception process constructs a proto-concept based on the interpreted and the anticipated information. The agent then uses action processes to affect the environment. Through comparing its anticipation with what actually occurs in the environment, the agent is able to evaluate the proto-concept.

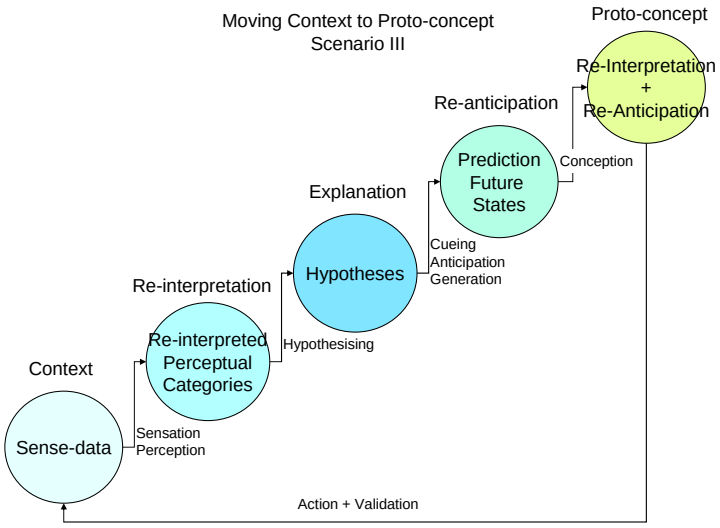

Figure 5. Moving a context into a proto-concept, Scenario III

Similar to Scenario I, in Scenario II the agent already has related experiences that can be used to construct a memory of a context. The difference between Scenarios I and II is timing. Scenario II describes that an agent reinterpreting a context when an invalid proto-concept is detected and discarded, as shown in Figure 4. Re-anticipations represent the agent's new predictions for the environment. It is produced through cueing and activating the memory. The constructed proto-concept can be regarded as an adaptive knowledge structure for a context over time. Scenario III, Figure 5, depicts how an agent comes up with a new concept when there is no mapping between the context detected and its experience.

Table 2. An example for Scenario III. Target concept is the concept to be learned (¬C means not C)

Context	**Time t**	**Time t + Δt**
Sense-data	User inputs a, b (symbols that represent features a, b)	User inputs a, b, ¬c, f (new combination of symbolic representation of features)
Agent's Experience	• a, b, c, ¬c, d, e, f, h, j, k correspond to A, B, C, ¬C, D, E, F, H, J, K (knowledge representations of symbols) • A, B, C, D, Target concept E (Experience of E and related representations of symbols) • A, B, C, F, Target concept J (Experience of J and related representations of symbols) • A, B, ¬C, F, H, J, Target concept K (Experience of K and related representations of symbols) • K ← ¬C (K cannot have C)	
Interpretation Reinterpretation	A and B	A, B, ¬C, F
Explanation	None	Not an E, Not a J, May be a K
Anticipation Re-anticipation	A, B, C, D, Target concept E	A, B, ¬C, F, H, J, Target concept K
Validity of Proto-concept	False	True

A new factor is included in Scenario III to provide explanations to compensate for the lack of mapping between the agent's experience and the context. The agent uses its experience to perform a reverse-engineering process, in which the context and interrelationships are analysed and represented in a new concept or the same concept with a different level of abstraction. A simple example is shown in Table 2.

An anticipatory system contains a predictive model of itself and its environment that enables it to adapt based on that model [34]. Anticipations are essential for an agent to be able to adapt to its environment. Anticipations bring forth the agent's experience and enable it to go beyond explicitly presented contexts. The interpretations and anticipations for a context form an instance of a proto-concept, which provides a snapshot of the agent's first-person representation of that context. The validity of the constructed proto-concepts can be evaluated through examining their anticipations in interactions. A valid proto-concept can correctly predict the environmental changes.

4.3 Modelling Experience of Situated Agents

Experience comprises the knowledge of some activity or some event gained through direct involvement in that activity or event. This paper represents experience as structures. They can be classified into three categories.

1. A 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. A perceptual experience captures historical representations of perceptual categories and their interrelationships, including entities, properties and entity–property relationships with degrees of beliefs;
3. A conceptual experience comprises the grounded invariants over the lower level perceptual experience. A conceptual experience explicitly states the regularities over the past observations of perceptual instances.

Experiential grounding [27] is implemented via a 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 [35], which is an extension of an IAC neural network [36], so that those nodes that fire together become more strongly connected.

5 SITUATED AGENT-BASED DESIGN OPTIMIZATION TOOL

This section describes the general architecture of a situated agent-based design optimization tool. A situated agent wraps around an existing design optimization tool. A user accesses a design tool via this wrapper, where the situated agent senses the events performed by that user. The situated agent uses its experience and concept formation engine to generate a concept, which modifies the tool's behaviour in later designing. The user can also directly communicate with the agent to obtain additional information. Such a framework provides the means that allow the agent to incrementally learn first-person knowledge. The system consists of two major components: a situated agent and a tool platform which includes a design optimization tool, a tool wrapper and interface agents, Figure 6. In this research, Matlab Optimization Toolbox (version 3.0.1) was chosen as the design optimization platform. It is a collection of functions that extend the capability of the MATLAB numeric computing environment (Release 14).

The toolbox includes routines for a variety of optimization classes, including unconstrained and constrained nonlinear minimization, quadratic and linear programming, and nonlinear optimization. Via the MATLAB command line, Matlab users use a scripting language called M-file to define and to solve optimization models. The interface agent, which consists of a Callback agent and an M-scripting agent, enables both users and the situated agent to operate on the engines in the Matlab Optimization Toolbox. A tool wrapper serves as an interface between the user, the tool and agents. It provides a simplified and efficient way to perform design optimization using the Matlab Optimization Toolbox.

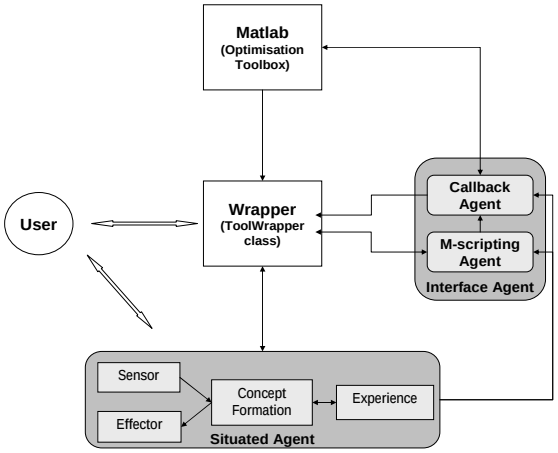

Figure 6. Situated agent-based design optimization tool (after Fig. 3. [8])

6 A COMPARISON EXPERIMENT IN DESIGN OPTIMIZATION

The purpose of this experiment is to evaluate the implemented prototype system through examining whether the proposed approach can learn new concepts from interactions, and evaluating the efficacy of the system in various design optimization scenarios. We measure the system's performance in assisting a design optimization tool to recognize novel design optimization problems compared to other approaches.

This test focuses on investigating the performance of various systems, namely a static system, a reactive system and a situated system, in learning to recognize design optimization problems 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;
- identifying objective function type;
- defining design variables, variable types;
- describing design constraints, constraint types;
- defining gradients of objective function and constraints;
- defining matrices, such as Hessian matrix and its type, A, b matrices (only available for Matlab users).
- selecting optimizers;
- submitting design problem or editing design problem; and
- submitting feedback on agent's outputs.

To support the test, a sequence of 15 design scenarios is created. The sequence of tasks is:

- {L, Q, Q, L, NL, Q, NL, L, L, NL, Q, Q, L, L, L}

"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.

A static system can only use the predefined knowledge to predict a design task. A reactive system uses *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:

$$P_s = \frac{Number\ of\ correct\ predictions}{Total\ numbers\ of\ predictions\ in\ the\ test} \qquad (7)$$

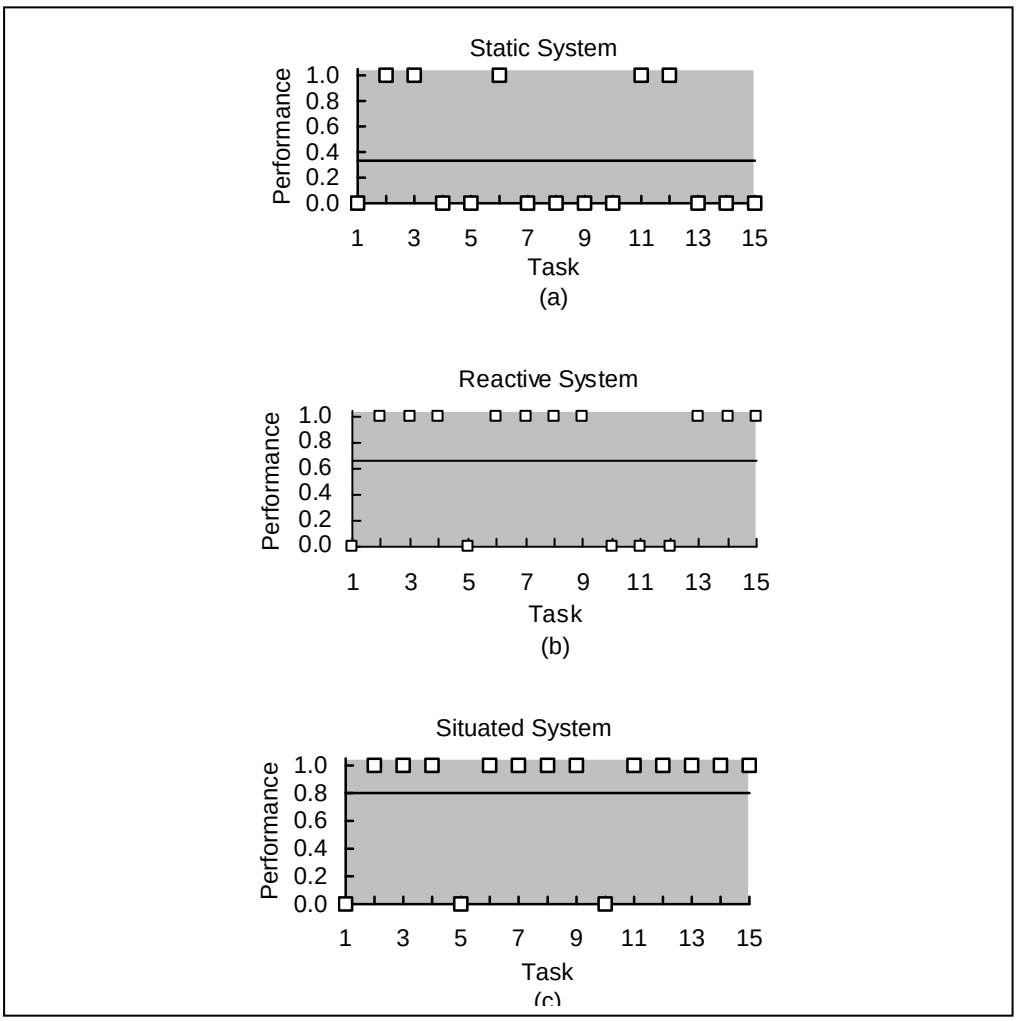

Figure 7. Prediction success rates for a static system (a); a reactive system (b) and a situated system (c) (white squares represent absolute results for predicting optimization problems, 1.0 is for correct predictions and 0.0 for otherwise)

The prediction success rate corresponds to the percentage of correctly predicted examples over total test examples. Based on results measured from this test, we can calculate prediction success rates for each system. As shown in the performance chart, Figure 7, a situated system produces a prediction success rate of 0.8 compared to a rate of 0.67 for a reactive system and 0.33 for a static system. We conjecture that 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 on.

7 CONCLUSION

This paper presents an approach that enables a design tool to learn first-person knowledge through interactions. A situated agent-based design optimization tool can learn to recognize new design optimization problems and adapt the learned concepts to various circumstances. Experimental results show that the proposed approach has a positive impact on improving efficacy of a design tool in assisting designers in their tasks. In conclusion, the approach plays a potential role in enhancing design effectiveness through introducing mechanisms that allow a design tool to learn first-person knowledge whist a designer is utilizing the tool in design.

REFERENCES

[1] Kalay, Y.E. The future of CAAD: From computer-aided design to computer-aided collaboration. In *Proceedings of the Eighth International Conference on Computer-Aided*

Architectural Design Futures, 1999, pp.14-39, (Kluwer Academic Publishers, Atlanta, Georgia).
[2] Gero, J.S. Design tools that learn: A possible CAD future. In Kumar, B., ed. *Information Processing in Civil and Structural Design*, 1996, pp.17-22 (Civil-Comp Press, Edinburgh).
[3] Gero, J.S. Design tools as situated agents that adapt to their use. In Dokonal, W. and Hirschberg, U., ed. *eCAADe21*, 2003, pp.177-180 (eCAADe, Graz University of Technology).
[4] Thomasson, A.L. First-person knowledge in phenomenology. In Smith, D.W. and Thomasson, A.L., ed. *Phenomenology and Philosophy of Mind*, 2005 (Oxford University Press, Oxford).
[5] Baker, L.R. First-person knowledge. In Sanford, A.J., ed. *The Nature and Limits of Human Understanding: The Gifford Lectures, University of Glasgow*, 2001, pp.165-184 (T.&T. Clark, London).
[6] Clancey, W. *Situated Cognition: On Human Knowledge and Computer Representations*, 1997 (Cambridge University Press, Cambridge).
[7] Peng, W. *A Design Interaction Tool that Adapts*. PhD Thesis, Key Centre of Design Computing and Cognition, 2006, 200pp. (University of Sydney, Sydney).
[8] Peng, W. and Gero, J.S. Concept formation in a design optimization tool. In Leeuwen, J.V. and Timmermans, H., ed. *Innovations in Design Decision Support Systems in Architecture and Urban Planning*, 2006, pp.293-308 (Springer, Berlin).
[9] Gero, J.S. Constructive memory in design thinking. In Goldschmidt, G. and Porter, W., ed. *Design Thinking Research Symposium: Design Representation*, 1999, pp.29-35 (MIT, Cambridge).
[10] Papalambros, P.Y. The optimization paradigm in engineering design: Promises and challenges, *Computer-aided Design*, 2002, 34, 939-951.
[11] Papalambros, P.Y. and Wilde, D.J. *Principles of Optimal Design: Modeling and Computation*, 2000 (Cambridge University Press, Cambridge, UK).
[12] Pardalos P.M. and Resende M.G.C. (Ed.) *Handbook of Applied Optimization*, 2002 (Oxford University Press, New York).
[13] Gero, J.S. and Kannengiesser, U. A framework for situated design optimization. In Leeuwen, J.V. and Timmermans, H., ed. *Innovations in Design Decision Support Systems in Architecture and Urban Planning*, 2006, pp.309-324 (Springer, Berlin).
[14] Balachandran, M.B. *A Model for Knowledge-Based Design Optimization*, PhD Thesis, 1988 (University of Sydney, Sydney).
[15] Hoeltzel, D.A. and Chieng, W.H. Factors that affect planning in a knowledge-based system for mechanical engineering design optimization with application to the design of mechanical power transmissions, *Engineering with Computers*, 1989, 5, 47-62.
[16] Reich, Y. and Fenves, S. The formation and use of abstract concepts in design. In Fisher, D., Pazzani, M. and Langley, P., ed. *Concept Formation: Knowledge and Experience in Unsupervised Learning*, 1991, pp.323-353 (Morgan Kaufmann, San Mateo, CA).
[17] Reich, Y. The development of BRIDGER: A methodological study of research in the use of machine learning in design, *Artificial Intelligence in Engineering*, 1993, 8(3), 165-181.
[18] Gero, J.S. Conceptual designing as a sequence of situated acts. In Smith, I., ed. *Artificial Intelligence in Structural Engineering*, 1998, pp.165-177 (Springer, Berlin).
[19] Radford, A.D. and Gero, J.S. *Design by Optimization in Architecture and Building*, 1988 (Van Nostrand Reinhold, New York).
[20] Dewey, J. The reflex arc concept in psychology, *Psychological Review*, 1896 reprinted in 1981, 3, 357-370.
[21] Bartlett, F.C. *Remembering: A Study in Experimental and Social Psychology*, 1932 reprinted in 1977 (Cambridge University Press, Cambridge).
[22] Suchman, L.A. *Plans and Situated Actions: The problem of human-machine communication*, 1987 (Cambridge University Press, Cambridge).
[23] Vygotsky, L.S. *Mind in Society: The Development of Higher Psychological Processes*, 1934 reprinted in 1978 (Harvard University Press, Cambridge, MA).
[24] Clancey, W. A tutorial on situated learning. In Self, J., ed. *Proceedings of the International Conference on Computers and Education*, 1995, pp.49-70 (Charlottesville, VA, AACE, Taiwan).
[25] Clancey, W. Review of Rosenfield's "The Invention of Memory". *Artificial Intelligence*, 1991,

50(2), 241-284.
[26] Gero, J.S. and Smith, G.J. A computational framework for concept formation for a situated design agent, Part B: Constructive memory, *Working Paper,* 2006 (Key Centre of Design Computing and Cognition, University of Sydney).
[27] Liew, P. and Gero, J.S. Constructive memory for situated agents. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, AIEDAM*, 2004, 18(2), 163-198.
[28] Nehaniv, C. and Dautenhahn, K. Embodiment and memories - Algebras of time and history for autobiographic agents. In *Proceedings of 14th European Meeting on Cybernetics and Systems Research, EMCSR'98*, 1998, pp.651-656 (Austrian Society for Cybernetic Studies).
[29] Harnad, S. The symbol grounding problem. *Physica D*, 1990, 42, 335-346.
[30] Ziemke, T. Rethinking grounding. In Riegler, A., Peschl, M. and Stein, A., ed. *Understanding Representations in the Cognitive Sciences*, 1999, pp.177-190 (Plenum Publisher, New York).
[31] Gero, J.S. and Kannengiesser, U. The situated function-behaviour-structure framework. *Design Studies*, 2004, 25(4), 373-391.
[32] Gero, J.S. and Fujii, H. A computational framework for concept formation in a situated design agent. *Knowledge-Based Systems*, 2000, 13(6), 361-368.
[33] Butterworth, G. Context and cognition in models of cognitive growth. In Butterworth, G. and Light, P., ed. *Context and Cognition: Ways of Learning and Knowing*, 1993, pp.1-13 (Lawrence Erlbaum Associates, Hillsdale, NJ).
[34] Rosen, R. *Anticipatory Systems*, 1985 (Pergamon Press).
[35] Peng, W. and Gero, J.S. Using a constructive interactive activation and competition neural network to construct a situated agent's experience. In Qiang, Y. and Webb, G., ed. *PRICAI 2006: Trends in Artificial Intelligence*, 2006, pp.21-30 (Springer).
[36] McClelland, J.L. Retrieving general and specific information from stored knowledge of specifics. In *Proceedings of the Third Annual Meeting of the Cognitive Science Society*, 1981, pp.170-172 (Erlbaum, Hillsdale, NJ).

Contact: Dr. Wei Peng
Commonwealth Scientific and Industrial Research Organisation (CSIRO) ICT Centre
Tasmanian ICT Centre
(*The Tasmanian ICT Centre 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 Development)*
Castray Esplanade, Hobart, TAS 7001
Australia
Tel: +61-3-6232-5536
Fax: +61-2-6232-5125
Email: wei.peng@csiro.au

Contact: Prof. John S Gero
George Mason University
Krasnow Institute for Advanced Study and Volgenau School of Information Technology and Engineering
Fairfax, VA 22030 USA
Tel: +1-703-415-6503
Email john@johngero.com
http://mason.gmu.edu/~jgero/