

## **INFORMATION AND COMMUNICATION TECHNOLOGIES IMPROVING EFFICIENCIES**

### **Case Study Paper**

## **TOWARDS A "LOOSELY-WIRED" DESIGN OPTIMIZATION TOOL**

**Wei Peng and John S. Gero**

*Key Centre of Design Computing and Cognition  
University of Sydney, NSW 2006, Australia  
[wpeng@arch.usyd.edu.au](mailto:wpeng@arch.usyd.edu.au)*

### **ABSTRACT**

The design optimization process involves a number of tasks that are both knowledge-intensive and error-prone. Most optimization tools focus on gathering a range of mathematical programming algorithms and providing the means for the user to solve design problems. Designers heavily rely on their experience to obtain optimal design solutions. This manual process can be arduous and inefficient. To improve the efficacy of the design optimization process, knowledge-based design optimization systems have been applied to provide knowledge support for tasks which require human expertise. These knowledge-intensive programs are hard-coded computer instructions that are not able to adapt to a dynamic design process. This paper describes learning mechanisms that allow design optimization tools to learn from their use – commencing as “loosely-wired” systems and “hard-wiring” themselves as they are used. A prototyped adaptive design optimization tool and its potential impacts are briefly described.

**Keywords:** Design optimization tool, knowledge, design process, situated agent, concept formation

## **1.0 INTRODUCTION**

The development of CAD (Computer-Aided Design) tools to support designing can be traced back to 1950s when the APT (Automatically Programmed Tool) was first launched at MIT. Computer-aided design tools, which emerged to assist designers in preparing drawings, specifications, and other design-related elements, now extend their dimensions to accommodate a vast variety of functionalities. Recently mathematical programming and optimization theory began to have a major impact on design. An optimal design can be obtained by solving an optimization problem. The design optimization process involves a number of tasks that are both knowledge-intensive and error-prone. Most optimization tools focus on gathering a range of mathematical programming algorithms and providing the means for the user to solve design problems. These design tools have invariably been built based on a paradigm that is founded on the notion that the tool is unchanged by its use (Gero, 2003). The knowledge and functions are encoded in what we call a "hard-wired" manner during the development stage. Designers rely heavily on their experience to obtain optimal design solutions. This manual process may result in sub-optimal design solution and hence inefficient design.

To improve the efficacy of a design optimization process, knowledge-based design optimization systems have been applied to provide knowledge support for tasks which require human expertise. These knowledge-intensive programs are hard-coded computer instructions that are not able to adapt to a dynamic design process. Motivated by a desire to build knowledgeable and personalized tools, a new research stream has emerged in the field of user modeling and interface agents. This includes work on the Lumiere project at Microsoft Research Centre (Horvitz et al., 1998), PBE systems (Lieberman, 2001) and interface agents (Maes, 1994) at MIT. Although these new tools take more proactive roles in assisting the user in some application domains, such systems are unable to adequately deal with dynamic situations that occur in designing.

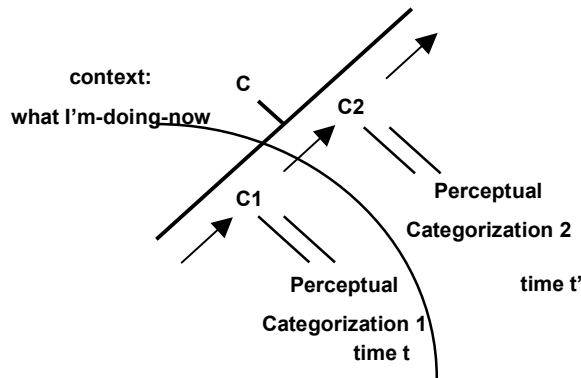
Design is a situated process in which designers interact with their design environments in developing the design (Gero, 1998). Interaction plays a critical role in shaping our design optimization practice in which similar design optimization problems may be solved in different ways. In order to assist designing in this dynamic process, it is necessary to address the interactions between the tool, the problem it is being used on and the user, in the sense that the tool is able to learn and adapt based on its experience to facilitate interactions. This paper describes learning mechanisms that allow design optimization tools to learn from their use – commencing as "loosely-wired" systems and "hard-wiring" themselves as they are used. A prototyped adaptive design optimization tool and its potential impacts are briefly presented.

## **2.0 SITUATED LEARNING PARADIGM**

Our approach is to utilise a situated agent to extend an existing design tool to model interactions, from which the agent is able to learn from its "experience". Via the agency provided, the tool is able to embody learning and to develop adaptive behaviour to assist designing. The paradigm on which the system depends to build new concepts from its interactions with its environment is founded on the ideas of "situatedness".

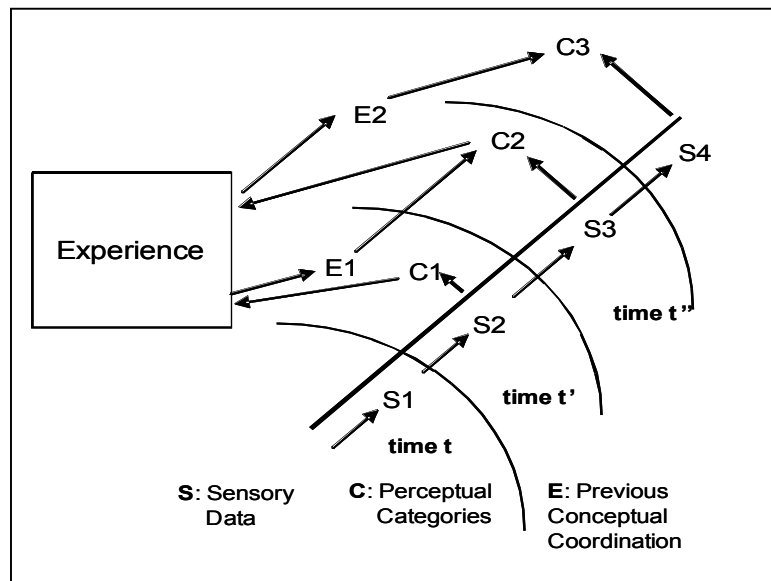
Situatedness involves both the context and the observer's experiences and the interactions between them. Situatedness is also referred as "where you are when you do what you do matters" (Gero, 1998). It states that an agent's knowledge depends on the context in which it is situated. Situatedness is inseparable from interactions in which knowledge is dynamically constructed as we conceive of what is happening to us, talk and move (Clancey, 1995). From this situated perspective, concept learning can be regarded as the way an agent orders its experience in time, which is proposed by Clancey (1999) as conceptual coordination. Conceptual coordination is the process where our everyday

experience is ordered by an ongoing understanding of what we are doing, where we are and what role we are playing in a larger social enterprise (Clancey, 1999), Figure 1.



**Figure 1.** Conceptual coordination (after Fig.1.6. of Clancey (1999))

A concept, which is a higher order categorization of a sequence, is generally formed by holding active a categorization that previously occurred (C1) and relating it to a currently active categorization C2, Figure 1. A concept is a function of previously organized perceptual categories and what subsequently occurs. Figure 2 illustrates a scenario of such a situated concept learning process in which sensory data is augmented into a Gestalt whole. Perceptual category C1 groups sensory sequence “S1→ S2” and activates the agent experience to obtain similar organizations. E1, as the agent’s experiential response, represents the agent’s hypotheses about what would happen in the environment at a later time. The agent constructs E1 with environmental changes (S3) into current perceptual category C2. This construction involves a validation process in which environmental changes are matched with the agent’s hypothesis. “Valid” means that the environmental changes are consistent with the agent’s projection of such changes from a previous time frame. The grounding process then reinforces a valid experience. For invalid expectations, the agent updates its perceptual category (C2) with the latest environmental changes. This incremental reflective process allows an agent to construct new concepts based on its previously conceptual coordination held in the experience.



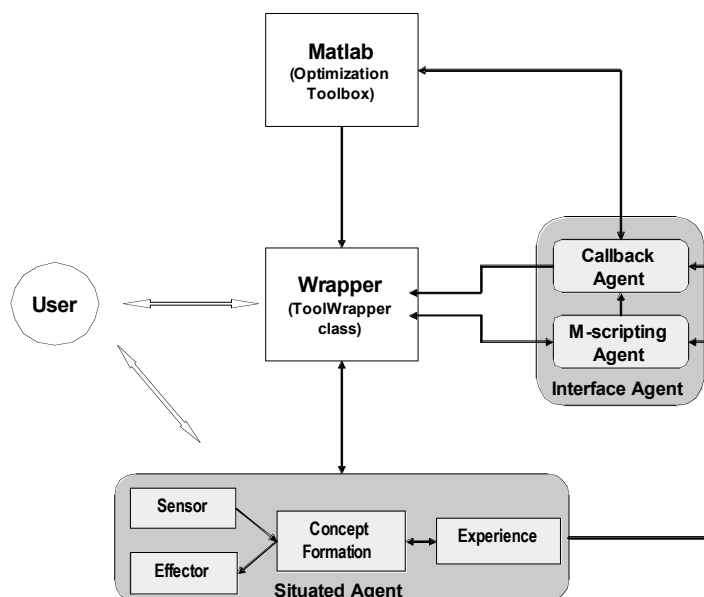
**Figure 2.** Situated concept learning processes

### 3.0 A SITUATED AGENT-BASED DESIGN OPTIMIZATION TOOL

How can a design optimization tool be developed as a situated agent? A wrapper entails a set of constructs that enable a tool to act as a computational rational agent, exhibiting autonomy independently of the functionalities it embodies (Gero, 2003). From sensor units that are embedded in the wrapper, the agent is able to gather a user's actions which are part of a design optimization process. These actions include key strokes of objective functions, the users' selections of design optimization algorithms, as well as gradients of objective functions, etc. These low-level sensory data are used by the situated agent to form concepts.

#### 3.1 THE ARCHITECTURE OF A SITUATED AGENT-BASED DESIGN OPTIMIZATION TOOL

Figure 3 shows the general architecture of a situated agent-based design optimization tool. The user accesses the design tool (Matlab Optimization Toolbox) via a wrapper, where a situated agent senses the events performed by that user. The situated agent uses its experience and concept formation engine to generate a concept, which changes the tool's behaviour. The user can also directly communicate with the agent to obtain additional information. Interface agents, which consist here of Callback agent and M-scripting agent, enable both users and the situated agent to operate on optimization algorithms in the Matlab Optimization Toolbox. Such a framework provides the means that allows the agent to incrementally learn new design experiences.



**Figure 3.** A situated agent-based design optimization tool that uses Matlab as the optimization tool

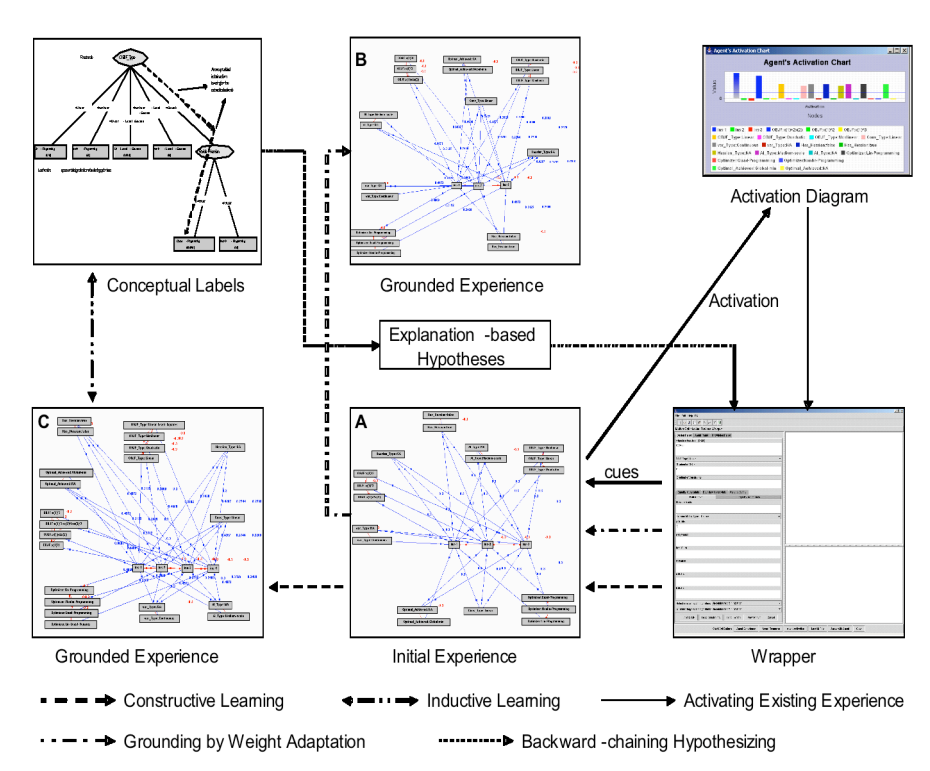
#### 3.2 THE SITUATED AGENT'S EXPERIENCE

The agent's experience is structured as two parts, those of organized conceptual instances and those of unstructured perceptual instances. Perceptual instance (P-Ins) refers to the experience that partially describes the instance of a design optimization problem. Conceptual instance (C-Ins) contains all necessary information of how a design optimization

problem is solved. It is composed of a number of perceptual instances. The conceptual instances are organized as a Constructive Interactive Activation and Competition (CIAC) neural network, in which we extend a basic IAC network (McClelland, 1981; 1995) to accommodate the concept learning process. An IAC has the ability to generalize across exemplars and to provide plausible default values for unknown variables<sup>1</sup>. Knowledge is extracted from the network by activating one or more of the nodes and then allowing the network to reach equilibrium (Medler, 1998). This organized experience changes in terms of weight adaptation and constructive learning as a result of interactions. Weight adaptation adjusts the weights of each excitatory connection so that those nodes that fired together become more strongly connected. Constructive learning incorporates new conceptual instances or reconfigures existing conceptual instances.

#### 4.0 THE PROTOTYPE SYSTEM

The implemented prototype system is illustrated in Figure 4. The tool wrapper interface allows designers to define problems. Embedded sensors gather a user's actions that comprise a design optimization process and activate a perceptor to create percepts.



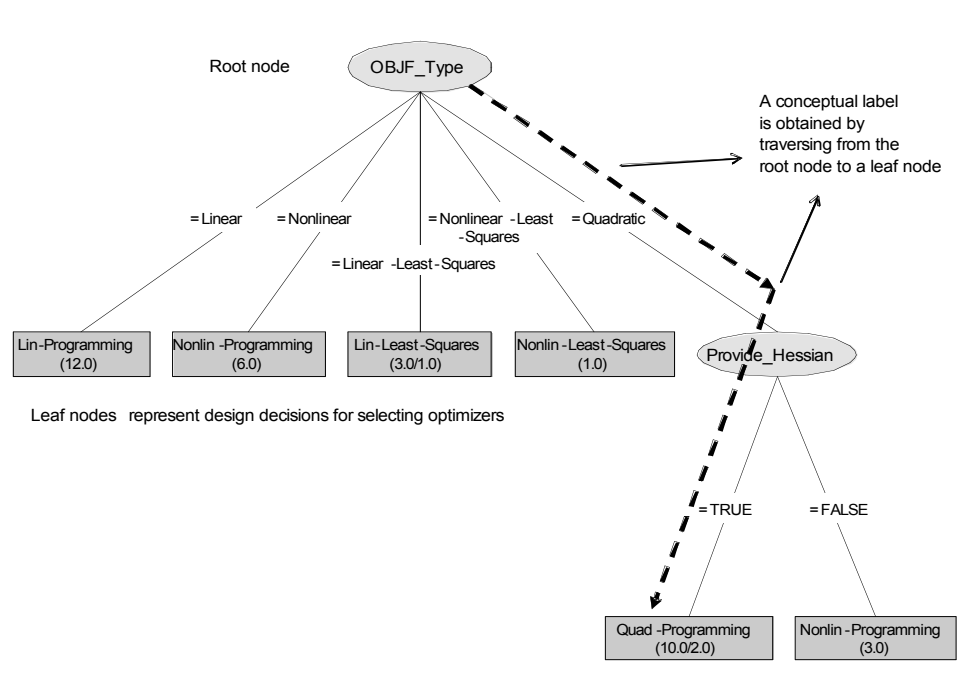
**Figure 4.** A prototype design optimization system that learns by its use

A percept cues the agent's initial experience. Activation diagrams output the neurons winning at the equilibrium state, which represent the activated memory. Based on the responses from a CIAC neural net, the agent constructs initial concepts and displays the constructed knowledge in the tool wrapper. Experiential grounding is the process that verifies the usefulness of a related experience in current situation (Liew, 2004). The grounding process initiates a validation function which matches the initially constructed concepts with environmental changes. Weight adaptation increases connection weights of the valid concept and grounds experience A to experience B. The explanation-based learner can be involved to form a new concept if no valid concept has been activated. A percept at

<sup>1</sup> <http://www.itee.uq.edu.au/~cogs2010/cmc/chapters/IAC/>

runtime can also be developed as a new concept by a constructive learning process. Experience C is learned from constructive learning and the related self conceptual labeling process. Conceptual labels are generalised knowledge that obtained from applying an inductive learner to the agent's experience.

Figure 5 shows the learning results and performance of applying a decision tree learner to the agent's experience. Each non-leaf node stands for a test on an attribute. Edges of the decision tree out of nodes are values of attributes for that node. Leaf nodes are used to represent design decisions for selecting optimizers. Numbers in parenthesis illustrate an observation for the class defined in the leaf node. For example, "3.0/1.0" describes that there are 3 positive observations and 1 negative observation for that class. A conceptual label can be obtained by traversing from the root node to a leaf node.



**Figure 5.** Conceptual labels learned from an inductive learner

As indicated in Figure 5, the agent has 83.3% (10 out of 12) confidence that a quadratic programming optimizer is suitable as an objective function is quadratic and a Hessian function is provided. A strong association is thus identified between quadratic objective function and Hessian matrix.

## 5.0 CONCLUSION

In summary, this paper introduced learning approaches that allow a design optimization tool to construct new concepts from interactions. The agent develops its structure and behaviour specific to what it is confronted with. Based on the conceptual knowledge learned, the agent can further improve the behaviour of the tool. As a result, designers can integrate their expertise with the knowledge learned from the agent to develop design solutions. A situated agent thus plays a potential role in supporting interactions in the design optimization process. Future research will focus on training and testing the implemented system.

## ACKNOWLEDGEMENT

This work is supported by a Cooperative Research Centre for Construction Innovation (CRC-CI) Scholarship and a University of Sydney Sesqui R and D grant. The research is carried out at Key Centre of Design Computing and Cognition, University of Sydney, Australia.

## REFERENCES

- Clancey, W (1995) A tutorial on situated learning, in J Self (eds), *Proceedings of the International Conference on Computers and Education*, Charlottesville, VA: AACE, Taiwan, pp. 49-70.
- Clancey, W (1999) *Conceptual Coordination: How the Mind Orders Experience in Time*, Lawrence Erlbaum Associates, New Jersey.
- Gero, JS (1998) Towards a model of designing which includes its situatedness, in H Grabowski, S Rude and G Grein (eds), *Universal Design Theory*, Shaker Verlag, Aachen, pp. 47-56.
- 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.
- Horvitz, E, Breese, J, Heckerman, D, Hovel, D and Rommelse, K (1998) The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, Madison, WI.
- 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*, PhD Thesis, University of Sydney, Australia.
- Maes, P (1994) Agents that reduce work and information overload, *Communications of the ACM* **37**: 31-40.
- 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.
- McClelland, JL (1995) Constructive memory and memory distortion: a parallel distributed processing approach, in DL Schacter (eds), *Memory Distortion: How Minds, Brains, and Societies Reconstruct the Past*, Harvard University Press, Cambridge, Massachusetts, pp. 69-90.
- Medler, DA (1998) A brief history of connectionism, *Neural Computing Surveys* **1**(1): 61-101.