Learning while optimizing a design: a situated agent-based design interaction tool

WEI PENG, JOHN S. GERO

Key Centre of Design Computing and Cognition, University of Sydney, Australia

Abstract

This paper presents an approach that enables a design tool to learn through its use. This approach uses a situated agent that wraps around a design tool and constructs concepts from interactions between the agent, the design problem and the use of the tool. The concept underpinning this research is founded on the ideas of "situatedness". Situatedness involves the context and the observer's experiences, as well as the interactions between them. A situated agent uses a constructive memory system to learn concepts while interacting. When applied to design optimization fields, such a situated agent-based design interaction tool demonstrates adaptive behaviours in reflexing, reacting and reflecting to various design optimization scenarios based on the incrementally learned knowledge structures and their intentional descriptions – the conceptual knowledge. Design tools no longer simply respond to their use based on *a priori* knowledge. Experiments show that a situated agent-based design optimization tool can learn and develop adaptive behaviours to support design decision-making. A prototype system is presented.

Keywords: Learning; Situatedness; Constructive Memory; Adaptive Behaviour; Design Optimization Tool

1. INTRODUCTION

Designing is recognized as one of the most complex human endeavours. Applying computers in building design can be traced back to the 1950s. Computer-aided design tools (CAD) were first introduced to assist designers in evaluating the "goodness" of their creations (Kalay, 1999). They have extended their functionality to include three-dimensional modelling, computer simulation, analysis, and integration between applications. Whilst the functionalities of design tools have been multiplying to embrace new technologies, e.g., gesture recognition, multimedia, and virtual reality (Myers, 1998), they continue to be built on the paradigm that the tool is unchanged by its use (Gero, 1996; 2003). Design tools keep repeating themselves irrespective of their interactions with the design environment. Designing is intrinsically dynamic and interactive, in the sense that designers reflect (Schon, 1983) and often change the course of the developing design (Gero, 1998a). Knowledge-based design systems have been widely applied to provide knowledge support for tasks which require human expertise, e.g., *OPTIMA* (Balachandran, 1988), *KNODES* (Rutherford & Maver, 1994) and *SEED* (Flemming, 1994). These systems encode sets of *a priori* design knowledge relating to the solutions of various design problems. To address the indeterministic nature of a design process, many CAD researchers turned to building systems that can learn automatically. Machine learning techniques have been widely adopted in knowledge-based systems to provide knowledge acquisition, modification and generalization. These systems apply a variety of machine learning algorithms to learn design product and process knowledge, e.g. design product learners like *ID3* (McLaughlin & Gero, 1987), *ECOBWEB* (Reich & Fenves, 1991), *BRIDGER* (Reich, 1993) and *NODES* (Persidis & Duffy, 1991; Duffy et al., 1995); design process learners such as *BOGART* (Mostow, 1989; Mostow et al., 1992) and *ARGO* (Huhns & Acosta, 1992). Classical design learning systems treat knowledge as universal applicable and context-free generalizations and descriptions (Reffat & Gero, 2000), so that they can be reused in different contexts.

It is worth noting that an equally important notion is "learning from a context". Lieberman and Selker (2000) pointed out that software that is sensitive to context can reduce users' frustrations in dealing with the complexity of their functionalities. The intrinsic interactive nature of design activities requires a system that integrates generalization and context, in the way that the system can learn and adapt to experiences in different circumstances.

Recently, interaction has been taken into account in developing systems to resolve uncertainty in a dynamic process. This includes research in the field of user modelling and intelligent interface, e.g.,

interface agents (Maes, 1994), the Lumiere project (Horvitz et al., 1998) and PBE systems (Lieberman, 2001). These adaptive interfaces model users and their interactions with computers with the aim of producing adaptive behaviours. From their initial endeavours based on the cognitive processes that drive users' actions in human computer interaction, research on user modelling now concentrates on learning users' habitual actions in using software applications. Adaptive interfaces provide proactive help in the sense that they anticipate user needs and present help before it is requested (Selker, 1994). However, adaptive interfaces have to cope with the "trade-off" between generalization and context (Lieberman & Selker, 2000). This creates a dichotomy between abstraction and context, which should be viewed as a coherent unity in what we call a *concept*.

In this paper, we present an approach founded on concepts from both the cognitive and computational domains. This approach uses a situated agent that wraps around a design tool and constructs concepts from interactions between the agent, the design problem and the use of the tool. As a combination of the agent's interpretations and expectations of its external and internal environment, a concept depicts the memory learned in a constructive memory model in a particular circumstance. A concept is a product of coordination of data-driven processes from contexts in the external world and expectation-driven processes from the agent's generalizations. It contains both the context-sensitive information and its grounded abstractions. A situated agent uses a constructive memory system to learn concepts while a designer is optimizing a design.

2. SITUATEDNESS AND CONSTRUCTIVE MEMORY

Software agents are intentional systems that work autonomously and interact with environments in selecting actions to achieve goals (Wooldridge & Jennings, 1995). A situated agent is a software agent built on the notion of "situatedness".

The concept of "situatedness" has its roots in empirical naturalism (Dewey, 1896 reprinted in 1981) and cognitive psychology (Bartlett, 1932 reprinted in 1977). It has been investigated in many different areas with diverse terms. It is also termed as "Situated Action" (Suchman, 1987), "Situated Cognition" (Clancey, 1997) and "Situated Learning" (Lave & Wenger, 1991). The notion of "situatedness" is considered as a *conditio sine qua non* for any form of "true" intelligence, natural or artificial (Lindblom & Ziemke, 2002). Vygotsky contributed to the concept of "situatedness" by introducing 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 (Vygotsky, 1978; Clancey, 1995).

The theory of situatedness claims that every human thought and action is adapted to the environment. They are situated because of what people perceive, how people conceive of their activity, and what people physically do develop together (Clancey, 1997). It is postulated in situated learning that knowledge is not a thing or set of descriptions or collection of facts and rules. Neither is it like procedures and semantic networks in a computer program. Human knowledge should be viewed as a capacity to coordinate and sequence behaviour, to adapt dynamically to changing circumstances (Clancey, 1995). In this vein, 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" (Gero, 1998b). It is inseparable from interactions in which knowledge is dynamically constructed. Situated cognition copes with the way humans construct their internal worlds via their interaction with the external world (Gero, 2003).

Central to the concepts of "situatedness" and a situated agent is the notion of "constructive memory". Constructive memory contradicts many views of knowledge as being unrelated to either its locus or application (Gero, 1998b). A memory can be regarded as a process of learning a concept. It is a reflection of how the system has adapted to its environment (Gero & Smith, 2006).

> *New memories of the experience are a function of both the original experience and previous memories of it. New memories can be viewed as new interpretations of the augmented experience* (Gero, 1999).

This process is biased by current environmental cues, experience and the way the agent's experiential responses are developed into memories from interactions between the agent and its environment. In this paper, we suggest that a constructive memory model can be implemented as a situated concept formation engine, through which a design interaction tool can learn.

3. SITUATED CONCEPT FORMATION

3.1. Concept Formation through a Situated Lens

Many researchers consider categorization the essence of a concept and its formation. Concept formation has

been regarded as a process of incremental unsupervised acquisition of categories and their intentional descriptions (Fisher & Pizzani, 1991). Based on this theory, a broad spectrum of computational models has been developed, including inductive learning methods, explanation-based learning approaches and connectionist algorithms. Theories of concept formation that merely focus on categorization are not able to address the complexity of the world (Bisbey & Trajkovski, 2005). A concept lacking an understanding of why the object, entity or event has its particular properties is called a protoconcept (Vygotsky, 1986; Bisbey & Trajkovski, 2005). We use the idea that learning a concept inherently involves understanding its influence on its environment. We believe that in the dynamic activity of designing, concepts that incrementally capture the knowledge of a design process are formed as a consequence of "situatedness" (Gero & Fujii, 2000; Peng & Gero, 2006).

Concepts are not only labelled structures, but also "multimodal categorizations of perceptual categorizations; ways of coordinating perception and action; meaning and activity are inseparable" (Clancey, 1997). It is the knowledge that enables an agent to reuse past situations in new situations (Gero & Fujii, 2000). We define concepts as the grounded invariants over the agent's experience. They are abstractions of experience that confer a predictive ability for new situations (Rosenstein & Cohen, 1998; Smith & Gero, 2000). A concept provides means for an agent to go beyond contexts and anticipate potential acts.

On the other hand, concepts are grounded in contexts. They are formed in the agent's interactions with the environment, and grounded as experiences when receiving positive feedback.

3.2. Conceptual Coordination

A concept formation process can be regarded as the way an agent orders its experience in time, which is referred to as conceptual coordination (Clancey, 1999). A concept is a function of previously organized perceptual categories and what subsequently occurs (see Fig. 1).

A concept is formed by holding active a categorization that previously occurred (C_1) and relating it to an active categorization (C_2) (Clancey, 1999). Fig. 2 illustrates a scenario of such a situated concept learning process. Perceptual category C_1 groups sensory sequence "$S_1 \rightarrow S_2$" and activates the agent's experience to obtain similar organizations. The agent's experiential response (E_1) represents the agent's expectations about what would happen later in the environment.

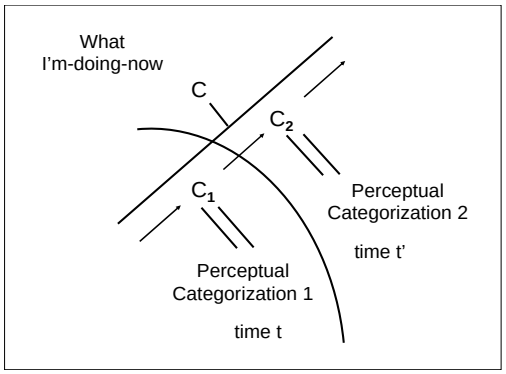

Fig. 1. Conceptual coordination shows the conceptual knowledge as a higher order categorization of a sequence (after Fig. 1.6 in Clancey (1999)).

The agent uses E_1 with environmental changes (S_3) to construct the current perceptual category C_2. This construction involves a validation process in which environmental changes are matched with the agent's expectation. "Valid" means the environmental changes are consistent with the agent's projection of such changes from a previous time. The grounding process then reinforces a valid experience. For invalid expectations, the agent updates its perceptual category (C_2) with the latest environmental changes. The concurrence of "situatedness" and "constructive memory" provides the basis for concept formation in a situated manner.

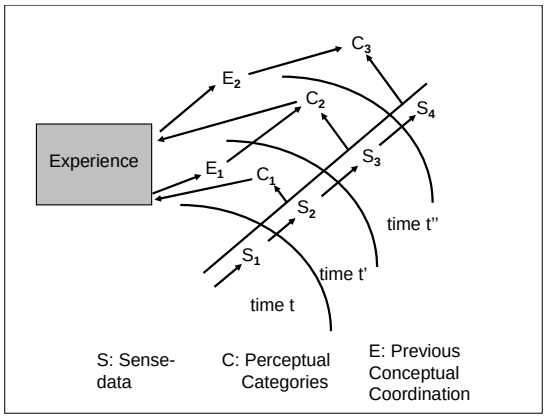

Fig. 2. A situated concept formation scenario.

4. A SITUATED AGENT-BASED DESIGN INTERACTION TOOL IN OPTIMIZATION

The system is applied in the design optimization domain. Design optimization is concerned with identifying optimal design solutions which meet design objectives while conforming to design constraints. The design optimization process involves some tasks that are both knowledge-intensive and

error-prone. Such tasks include problem formulation, algorithm selection and the use of heuristics to improve efficiency of the optimization process. Designers rely on the experience that they have built up through years of practice to carry out these tasks. The outcome of this design process is constrained by their expertise. Design knowledge plays a critical role in improving the efficiency and efficacy of this process. We concentrate on illustrating a scenario of how the proposed model contributes to a design optimization process.

4.1. Application Scenario

A large number of optimization algorithms have been developed and are commercially available. 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. For example, Matlab Optimization Toolbox 3.0 [1] includes a variety of functions for linear programming, quadratic programming, nonlinear optimization and nonlinear least squares, etc. Choosing a suitable optimizer becomes the bottleneck in a design optimization process. The recognition of appropriate optimization models is fundamental to design decision problems (Radford & Gero, 1988). Some of the knowledge required for 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 therefore offer help to designers in their interactions in designing even before they require it. The design tool that adapts based on its experience of its use is claimed to be effective (Gero, 2003). The effectiveness of the tool describes the capability of producing an effect.[2] It is often associated with the term "efficacy" when a design tool is applied in a design activity. The efficacy of the tool concentrates on the outcome and usually refers to the ability to produce a desired amount of a desired effect.[3] The tool that maintains 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. In a design optimization scenario, it can be measured through the correctness in recognising a design optimization problem.

Table 1. *An example of knowledge required in recognition of an optimization problem (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

4.2. Architecture of the Interaction Tool in Optimization

Fig. 3 demonstrates a general architecture of this situated agent-based design interaction tool used in optimization. A situated agent wraps around an existing design optimization tool. A user accesses a design tool via a wrapper, whereby the situated agent senses the events performed by that user.

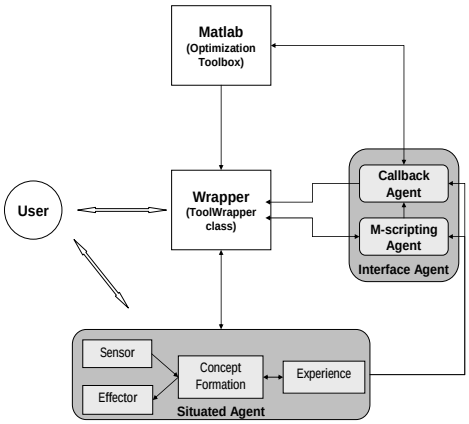

Fig. 3. A general architecture of a situated agent-based design interaction tool in optimization.

The situated agent uses its experience and concept formation engine to generate a concept, which affects the tool's behaviour in designing. The user can also directly communicate with the agent to obtain additional information. Such a framework provides the means that allows the agent to incrementally learn new experiences. The system consists of two major components: a situated agent and a tool platform

[1] http://www.mathworks.com/products/optimisation/.
[2] http://en.wikipedia.org/wiki/Effectiveness.
[3] http://en.wikipedia.org/wiki/Effectiveness.

which includes a design optimization tool, a tool wrapper and interface agents.

4.3. Design Optimization Tool Platform

In this paper, the Matlab Optimization Toolbox (version 3.0.1) is 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.

4.4. A Situated Agent

A situated agent therefore 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. These events include users' actions performed in using a design tool, such as key strokes, menu selections and mouse clicks. Sense-data are environment variables and their states that are captured by sensors. They are triggered by external environmental changes (exogenous). Exogenous sense-data (S_e) take the form of a sequence of actions. For example, some sense-data at time "t" can be expressed as:

- S_e (t) {…… "click on a certain text field", key stroke of "x", "(", "1", ")", "+", "x", "(, "2", ")"…}.

These sense-data are transferred into sensory data which are grouped into categories according to their initial descriptions. 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 produced from matching the agent's exogenous sense-data (S_e) with the agent's sensory level experience. Sensory data (S_{e+a}) are a combination of the agent's exogenous sense-data (S_e) and the related autogenous information (S_a).

For example, these sense-data can be converted into a sequence of labelled sensory data at time "t":

- S_{e+a} (t) {…… "text field label: key stroke of "x", "(", "1", ")", "+", "x", "(, "2", ")""…}.

Percepts are intermediate data structures that are generated from mapping sensory data into categories. The perceptor processes sensory data and groups them into multimodal percepts. Percepts are structured as triplets:

- P (Object, Property, Values of properties).

For example, perceptual data at time "t" (P (t)) built on the above sensory data can be described as:

- P (t) { P_1, Label for Text Field, "x(1)+x(2)"}.

Percepts can also be grouped into sequences of categories. These are composed percepts. A cue_Maker discerns whether a percept can be used to create a memory cue. This is performed by matching with the agent's sensory-level experience. The cue_Maker is also responsible for creating cues that can be used to activate an agent's context-addressable experience.

The conceptor is the software object used to generate concepts in the conception process. A concept is regarded as a result of an interaction process in which meanings are attached to a perceptual category. 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[4] about its external and internal environment at a particular time. 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 cue its experience structure to generate anticipations for the interpreted perceptual category. The conception process constructs a proto-concept based on the interpreted and the anticipated information.

The conception process 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 environmental cue. This includes deriving anticipations from these responses and

[4] The interpretation is concerned with associating an initial meaning to a context. Anticipation is the concept of an agent making decisions based on predictions and expectations about the future. http://en.wikipedia.org/wiki/Anticipation.

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 experience structures.

The validator pulls information from the environment and examines whether the environmental changes are consistent with the agent's anticipations. The hypothesizer generates hypotheses from the learned proto-concepts when they are not valid in interactions. This 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 theory (usually conceptual experience). An agent subsequently refocuses on or constructs a new proto-concept based on hypotheses. An effector is the unit via which the agent brings changes to environments through its actions.

4.5. The Agent's Experience

The agent's experience is treated as knowledge structures that hold sensory, perceptual and conceptual representations and can be used to construct a memory. They can be classified into three categories.

1. 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 environmental stimulus is detected, the agent attaches an initial meaning to it, based on its sensory experience.
2. Perceptual experience captures historical representations of perceptual categories and their interrelationships. These include entities, properties and entity–property relationships with degrees of beliefs.
3. 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. It is obtained via grounding the concepts formed from interactions.

4.5.1. The agent's sensory and perceptual experience

The sensory experience contains symbolic labels for each sensory category. Based on this level of experience which holds modality information, the sense-data can be transformed into discrete perceptual categories. Sensory experiences are labels that are kept consistent with their higher level organizations – the perceptual experience. For instance, the name of "OBJF_Type" denotes a sensory experience for a category "Objective Function Type".

Perceptual experience is organized into a Constructive Interactive Activation and Competition (CIAC) neural network, which is an extension of a basic IAC network (McClelland, 1981; 1995). An IAC network consists of two basic nodes: instance nodes and property nodes. The instance node has inhibitory connections to other instance nodes and excitatory connections to the relevant property nodes. The property nodes encode the special characteristics of an individual instance (Medler, 1998). Property nodes are grouped into cohorts of mutually exclusive values. Each property node represents the perceptual level experience which is processed from sensory data. Instance nodes along with the related property nodes describe an instance of a concept. Knowledge is extracted from the network by activating one or more of the nodes and then allowing the excitation and inhibition processes to reach equilibrium (Medler, 1998).

As shown in Fig. 4, the shaded instance node (❶) and related shaded property nodes (■) present a context addressable memory cued from an environmental stimulus, e.g., [Objective_Function, f_1]. Such knowledge is a dynamic construction in the sense that the agent develops adapted experience as environmental stimuli change.

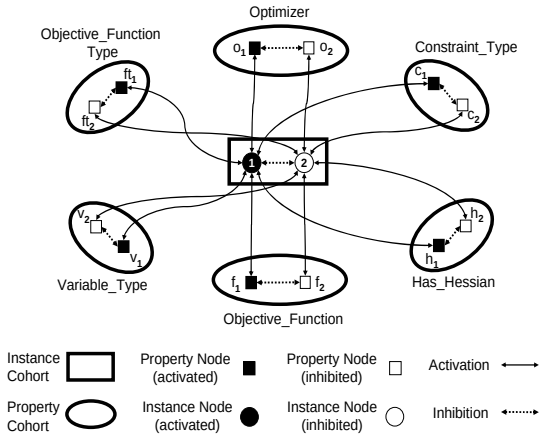

Fig. 4. A CIAC neural network as a representation of the agent's experience; Property nodes are labelled by their value, e.g., f_1 represents a property node in the Objective_Function cohort with objective function value f_1.

An activation diagram (Fig. 5) outputs the neurons winning at the equilibrium state, which represent the activated memory. The horizontal line shows the activation threshold, which is an empirical value defining the boundary of the activation. Only those neurons that surpass this value yield outputs.

Based on the responses from a CIAC neural network, the agent constructs initial concepts and displays the constructed knowledge in the tool wrapper.

Table 2 illustrates the formulae that are used to compute network input value (N_e), activation value (A_c) for each node during the activation and competition phases.

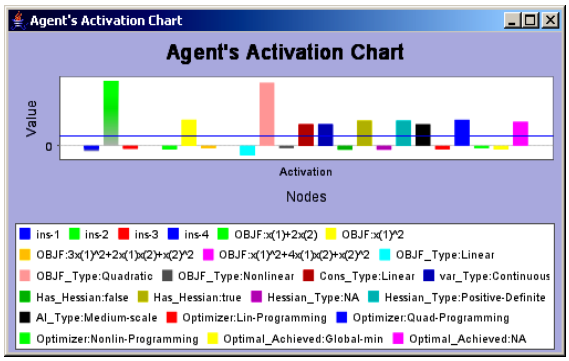

Fig. 5. Activation diagrams show the activated memory of design experience 2 ("ins-2" stands for instance node 2, which is connected to a number of property nodes, i.e. "OBJF:x(1)^2", "OBJF_Type: Quadratic", ……, "Optimizer: Quad-Programming").

Table 2 also describes the formula that is applied to adjust the weights of each excitatory connection of the valid concept during the grounding via weight adaptation process, so that those nodes that fired together become more strongly connected. Weight adaptation is formulated similar to a Hebbian-like learning mechanism (Medler, 1998).

We use the response value R_a to measure the strength of an experience. It is specified as follows:

$$R_a = \sum_{i=1}^{n} A_i \qquad (1)$$

where n is the number of nodes in the network and A_i denotes an activation value of a node. R_a is defined as the sum of activation values for the neurons at a specific time. We utilize activation gain ΔA to measure the increase of activation value for each node for two consecutive cycles. Formula (2) illustrates this variable.

$$\Delta A = A_i - A_j \qquad (2)$$

where A_i, A_j represents the activation values of a node at two successive cycles. Therefore, the equilibrium state refers to the state in which no more activation gain (ΔA is less than a threshold) is acquired by the network activation and competition process.

The response diagram shows how an experience is triggered during cycles of activation and competition phases, Fig. 6, where the X coordinate represents the cycle number, which is a time variable.

Table 2. *Major formulae applied in CIAC neural network. A_c represents the strength of belief of the activated node; weight adaptation is a grounding process that verifies the usefulness of a related experience in current situation.*

Item	Formulae
Network Input Values (N_e)	$N_e = \mu E + \sum_{i=1}^{n} W_i A_i$ μ: excitatory gain for initial network stimulus, set to 4.0 E: initial network stimulus, default 1.0 W_i: inbound weights for a neuron A_i: activation value for each inbound weight n: number of neurons in a network
Activation Values (A_c)	*If* $N_e > 0$ $A_c = A_{c-1} + \ell[(A_{max} - A_{c-1})N_e - \varphi(A_{c-1} - R)]$ *else* $A_c = A_{c-1} + \ell[(A_{c-1} - A_{min})N_e - \varphi(A_{c-1} - R)]$ A_c: activation value for node at current cycle A_{c-1}: activation value for node at previous cycle N_e: net input for each node A_{max}: maximum activation value, set to 1.0 A_{min}: minimum activation value, set to -0.2 R: initial resting activation value, default -0.1 φ: the decay factor, default 1.0 ℓ: the learning rate, default 0.1;
Weight Adaptation (W_n):	*If* $W_o > 0$ $W_n = W_o + \ell(W_{max} - W_o)A_i A_j - \delta W_o$ *else* $W_n = W_o$ W_o: weight value before weight-adaptation W_n: weight value after weight-adaptation ℓ: learning rate, default 0.1 W_{max}: maximum weight value, set to 1 A_i, A_j: activation values for neuron i and j δ: weight decay factor, set to 0.1

The Y coordinate describes the response value (for the ■ curve) and the sum of activation gain ΔA of the experience (for the ● line).

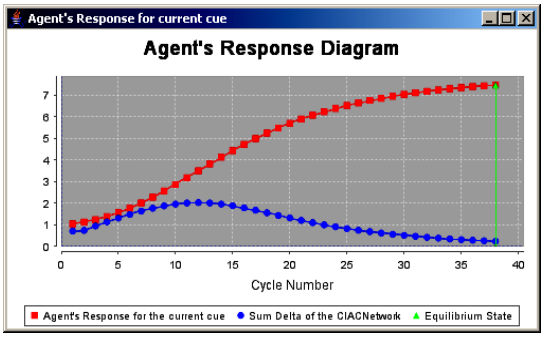

Fig. 6. The response diagram shows the dynamics of the activation and competition processes.

4.5.2. The agent's conceptual experience

The agent's conceptual experience can be represented in a decision-tree structure. Fig. 7 shows the learning results and the performance of applying a decision

tree learner (in the induction function of the conception process) to the agent's experience. Decision tree learning is a widely used method for inductive inference. Each non-leaf node stands for a test on an attribute. Edges of the decision tree coming from the nodes are values of attributes for that node. Leaf nodes are used to represent design decisions for selecting optimizers. Numbers in parentheses illustrate an observation for the class defined in the leaf node. For example, "5/1" describes that there are five positive observations and one negative observation for that class. A conceptual knowledge (or label) can be obtained by traversing from the root node to a leaf node.

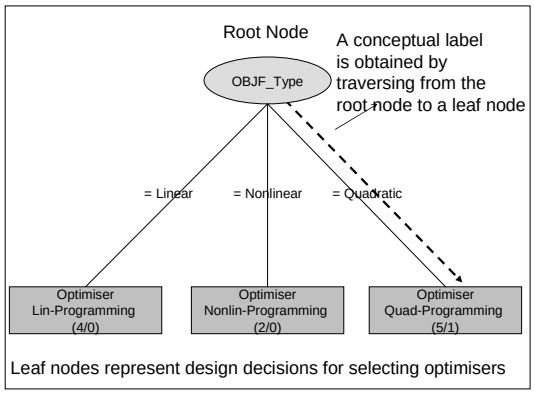

Fig. 7. A decision tree learned from C4.5.

The agent's conceptual experience serves as domain theories for the agent's hypothesizer to construct explanations. Fig. 8 illustrates a hypothesis created on the basis of the conceptual knowledge obtained in Fig. 7. The hypothesis that is learned from a deductive learner allows the agent to reinterpret and re-anticipate a circumstance.

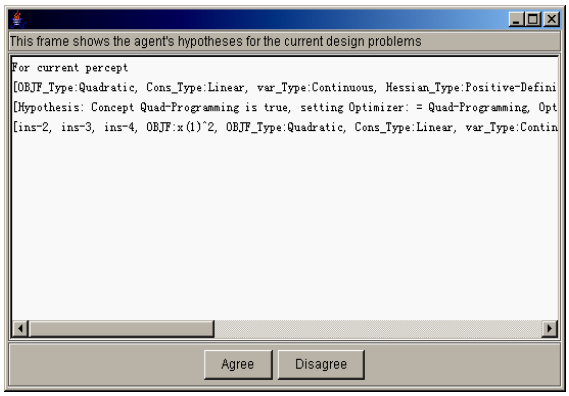

Fig. 8. A hypothesis and the related proto-concept.

4.6. Experiential Grounding

Symbolic grounding explores the means by which the semantic interpretation of a formal symbol system can be made intrinsic to that system rather than relying on the meanings in the head of an external interpreter or observer (Harnad, 1990). The grounding problem generally refers to representation grounding (Chalmers, 1992) or grounding of concepts, in which concepts can be developed through interactive behaviour in an environment (Dorffner & Prem, 1993). An agent grounds its behaviour and representations in its interaction with the environment. The behaviour of the agent is intrinsically meaningful to itself (Ziemke, 1999). The key issue leading to a truly grounded AI system is "getting there", in the sense that the agent could construct and self-organize itself and its own environmental embeddings (Ziemke, 1999).

In this vein, in implementing a constructive memory system, "experiential grounding" (Liew & Gero, 2002) 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 as being reactive in current time. In this paper, a grounding process is referred to as the testing and the evaluation of whether a constructed memory correctly predicts environmental changes. A function called "weight adaptation" (also in Table 2) is applied here. Fig. 9 (a) shows an experience structure before the grounding process.

This structure contains two design instance nodes, among which "ins-2" denotes an instance of a quadratic design optimization experience. The two instance nodes ("ins-1" and "ins-2") are shown in the centre of the network. They are surrounded by and connected with some property nodes. The label inside a property node shows the cohort in which a node is located and the value of that property node. For instance, the label "OBJF_Type: Quadratic" describes an "OBJF_Type" property node with value "Quadratic". The weighted connections between the instance node ("ins-2") and property nodes "OBJF_Type: Quadratic" and "Optimizer: Quad-Programming" are both "0.2942" (circled by dotted lines in Fig. 9 (a)). Fig. 9 (b) presents the grounded experience. It is noted that the connection weights between instance node ("ins-2") and property nodes "OBJF_Type: Quadratic" and "Optimizer: Quad-Programming" are enhanced to "0.3655" and "0.3155" respectively. The amounts increased are not only decided by the weights before the grounding process, they are also influenced by the activation values in the state of equilibrium.

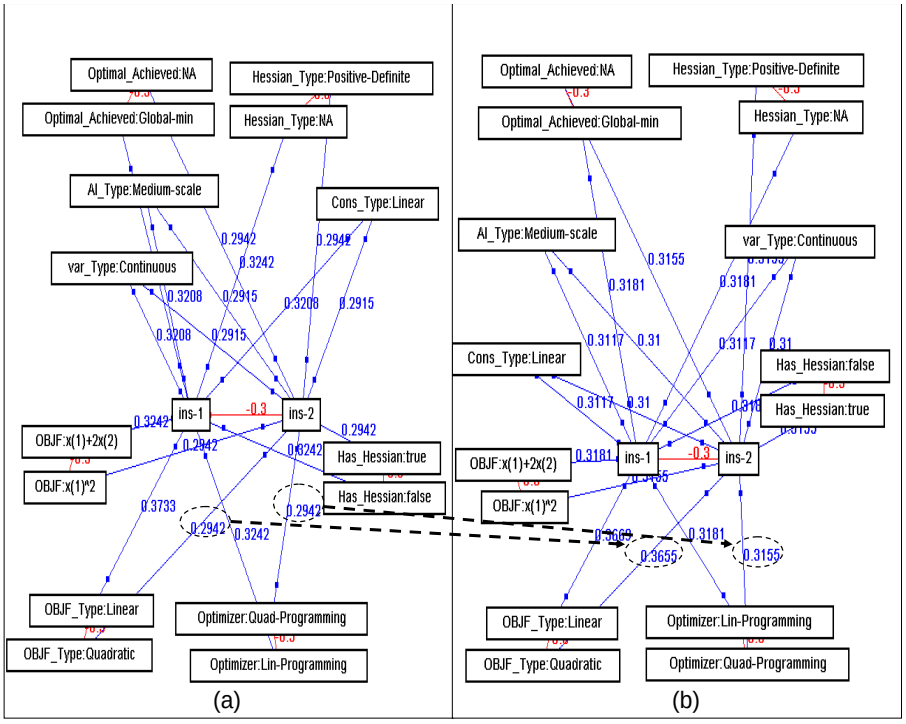

Fig. 9. Grounding via weight adaptation changes the connections between neurons according to their usefulness in interaction.

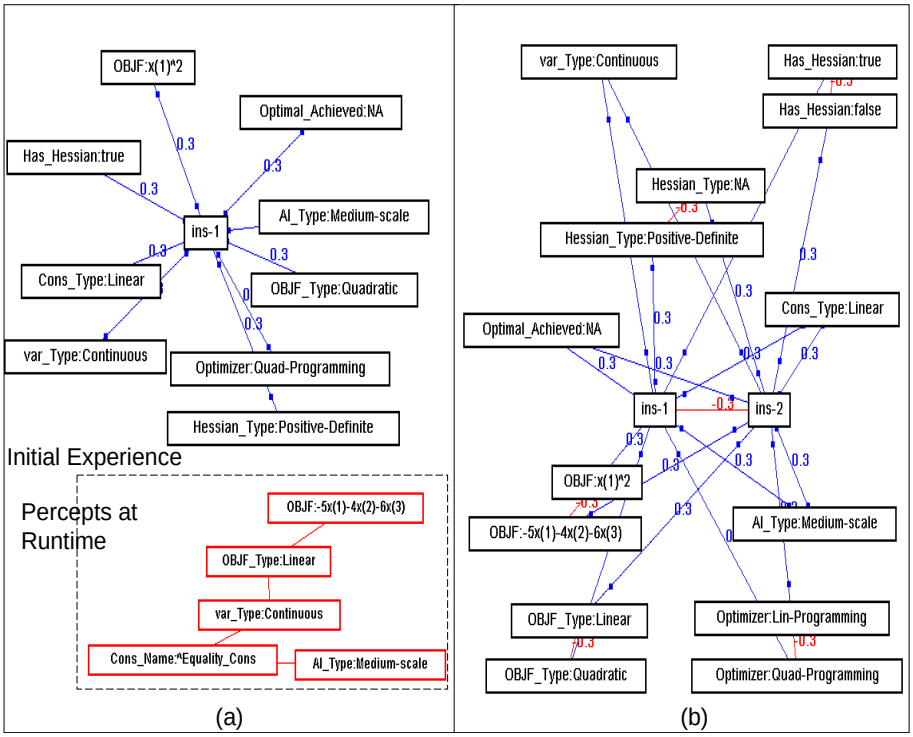

Fig. 10. A new experience is learned from perceptual categories at run-time.

Another process that contributes to grounding is the constructive learning mechanism (Liew & Gero, 2002) of a constructive memory system. This allows an agent to accommodate a new experience in an *a posteriori* manner. Constructive learning (C-Learning) has been implemented and illustrated in Fig. 10. Fig. 10 (a) describes an initial experience structure and the percepts at run-time. The agent learns a new experience based on these percepts. The new instance node is depicted as "ins-2" in Fig. 10 (b).

4.7. Adaptive Behaviour

Adaptivity, the agent's capability to learn and improve with experience (Bradshaw, 1996), is a significant property of a situated agent. Adaptation is not only concerned with learning new knowledge structures, it also refers to self-organizing the system's knowledge and behaviours in the interactions. Adaptation refers to adjustments of a situated agent's behaviours in its interaction with the environment. Adaptive behaviours, in terms of reflexive, reactive and reflective behaviour (Maher & Gero, 2002), can result from the agent's internal processes and constraints imposed by a situated agent architecture.

An agent reflexes when its experiential response to the current memory cue is sufficiently strong to directly influence the action without reasoning. This can be a result of highly grounded experience, i.e., the weighted connection between the node representing an environmental cue and another experience node reaches a certain threshold. In reactive behaviour, the agent is able to activate an existing experience on environmental cues. The knowledge structures (CIAC network) of the agent's experience are involved in producing memory, which is anticipation for the potential consequence of environmental changes. The agent is able to evaluate the experience of that memory by pulling environmental changes and comparing these changes with the activated memory.

The agent's reflective mode is triggered by discrepancies between the agent's anticipation and current environmental changes. There are two scenarios of reflective behaviours. In reflective behaviour scenario I, a memory is reactivated when the agent's initial experiential response to a memory cue fails in the validation process, but a memory cue is still able to be subsequently identified in the environment. In reflective behaviour scenario II, a deductive learner is employed to create hypotheses, which are based on the explicitly represented domain theories and its goal concept. The agent therefore refocuses to an existing concept or initiates constructive learning.

5. THE ARCHITECTURE OF THE IMPLEMENTED SITUATED AGENT

The architecture of the implemented situated agent is illustrated in Fig. 11. The tool wrapper interface allows designers to define problems. Sensors gather a user's actions that comprise a design optimization process and activate a perceptor to create percepts. A percept cues the agent's initial experience. Based on the responses from the CIAC neural network, the agent constructs initial concepts and displays the constructed knowledge in the tool wrapper (solid lines in Fig. 11).

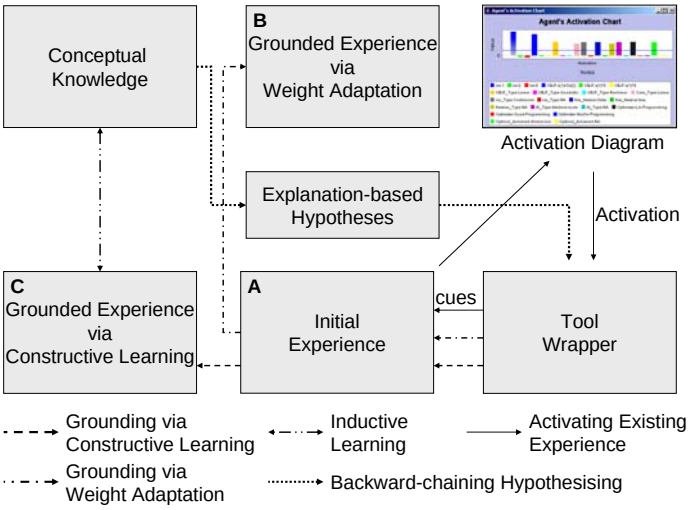

Fig. 11. The architecture of the implemented situated agent.

The grounding process initiates a validation function that matches the initially constructed concepts with environmental changes. Weight adaptation increases the connection weights of the valid concept and grounds Experience A to Experience B (dash dot lines in Fig. 11). In the agent's reflective concept learning process, the explanation-based learner is used to form a new concept (square dot lines in Fig. 11). A percept at run-time can also be developed as a new concept by a constructive learning process (dash lines in Fig. 11). Experience C is learned using constructive learning and the related conception process.

Typical scenarios of experiential grounding have already been demonstrated in Fig. 9 and Fig. 10, e.g., grounding of Experience A to B can be viewed in Fig. 9 (a) and Fig. 9 (b). Similarly, the grounding via the constructive learning process that transfers Experience A to Experience C can be found in Fig. 10 (a) and Fig. 10 (b). Conceptual knowledge and explanation-based hypotheses shown in Fig. 11 have also been described in Fig. 7 and Fig. 8.

6. EXPERIMENTS AND RESULTS

In this section, we describe a number of experiments and discuss their results. The purpose of the experiments is to evaluate the proposed interaction tool through:

- examining whether the proposed approach 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;
- studying the characteristics of the agent's behaviours in various circumstances;
- evaluating the efficacy of the interaction tool. We measure the system's performance in assisting a design optimization tool to recognise novel design optimization problems compared to other approaches.

Experiment I is concerned with investigating how a situated agent develops knowledge structures and behaviours in similar design optimization scenarios over time. It measures the agent's response value (R_a) and response time (T_e) in a number of linear design optimization scenarios. Experiments II and III focus on observing and analysing the tool's behaviours in heterogeneous design optimization scenarios. Experiment IV is a comparative test of various systems and their performance in learning to recognize novel design optimization scenarios.

6.1. Experiment I

In this experiment, the initial agent holds a linear design optimization experience, which is represented as a CIAC neural network that contains one instance node and related nine property nodes. This is a structure that can be used to construct a memory of a linear optimization problem. The learning rate of this agent is set to 0.205 and the threshold of equilibrium state (T_{eq}) is 0.005. Each time the agent responds to a memory cue "OBJF_Type:Linear", it subsequently grounds the constructed memory into a new experience.

Fig. 12 shows a network view of such an initial experience. Table 3 gathers the agent's states in 10 consecutive testing epochs on linear design optimization tasks. Response value R_a represents the sum of the activation value for each node of the CIAC neural network. The response time of the agent T_e can be obtained by counting computer cycles for a CIAC network to reach the equilibrium state. "Mean Activation" is the average activation value for a CIAC network. "Sum of Δa" depicts the sum of the activation gains (the increase of an activation value for each node for two consecutive cycles) for each node in the state of equilibrium. Weights of connections of the CIAC network are also recorded in a matrix file. The performance of the agent is defined as the prediction correctness.

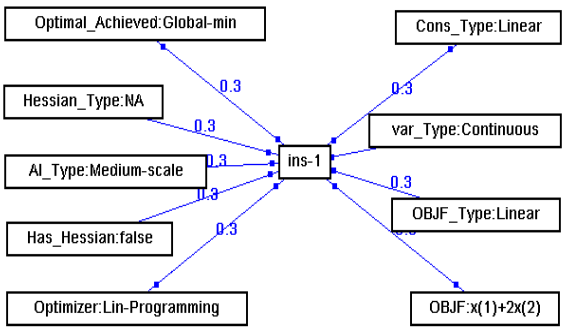

Fig. 12. The initial experience structure has one instance node which is connected to nine property nodes with connection weight 0.3. It denotes a scenario of linear design optimization.

Fig. 13 illustrates how the agent modifies its behaviours based on its experience. Exposed to similar design problems, the agent improves its responses based on the adaptation of its knowledge structures. The strength of its experience is related to the time (T_e) and response value (R_a) with which the agent responds to an environmental cue.

Table 3. *Grounding of the agent's experience in similar design optimization scenarios.*

Testing Epoch	Response (R_a)	Time to Equilibrium (T_e)	Mean Activation	Sum of Δa	Performance (0.0 – 1.0)	Weighted Connection (0.3000 – 1.0000)
1	6.891	44	0.6981	0.2366	1.0	0.3000
2	7.478	39	0.7478	0.2283	1.0	0.3899
3	7.870	34	0.7870	0.2199	1.0	0.4616
4	8.130	31	0.8130	0.2155	1.0	0.5373
5	8.281	28	0.8281	0.2402	1.0	0.6053
6	8.429	27	0.8429	0.2183	1.0	0.6593
7	8.529	26	0.8529	0.2101	1.0	0.6981
8	8.596	25	0.8596	0.2119	1.0	0.7394
9	8.639	25	0.8639	0.2237	1.0	0.7657
10	8.696	25	0.8696	0.2007	1.0	0.7910

In Experiment I, the experience of the system is enhanced by grounding via a weight-adaptation process.

As illustrated in the results, the experience gained in solving similar design problems is enhanced with its response values increased and response time reduced over time. The tool recognizes similar design situations with a high accuracy and an improved response rate.

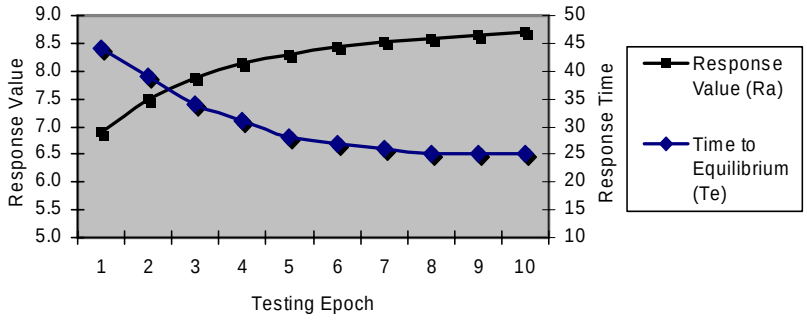

Fig. 13. The agent adapts its experience in Experiment I.

6.2. Experiments II and III

Each experiment uses a sequence of simulated design scenarios. 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 an objective function;
- identifying the objective function type;
- defining design variables, variable types;
- describing design constraints, constraint types;
- typing in the gradients of objective function and constraints;
- defining matrices, such as Hessian matrix and its type;
- selecting optimizers;
- submitting a design problem or editing a design problem;
- submitting feedback on the agent's outputs.

To support Experiment II, 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. Table 4 shows the symbols used to represent behaviours of the agent. The following seven internal states are recorded and illustrated in Table 5:

1. The knowledge structure is represented in a Constructive Interactive Activation and Competition (CIAC) neural network composed of instance nodes connecting to a number of property (or feature) nodes;
2. The expectations about environmental changes are generated by the agent's experiential responses to environmental cues (A_c);
3. The validator states show whether an agent's expectation is consistent with the environmental changes (V_1 and V_2);
4. The reactivated experience or initially validated experience is experience reactivated during the reflective learning process or the validated experience during validation (R_c);
5. Hypotheses depict the agent's reinterpretation about its failures in creating a valid expectation (H_s);
6. Concepts are the agent's high-level experiences, which are domain theories an agent uses to classify and explain its observations (N_k);
7. The directly observed system's behaviours are in terms of "sensation", "perception", "conception 1-3", "IAC neural network activation", "IAC neural network reactivation", "reflexive experience response", "hypothesizing", "validation", and "grounding via weight adaptation". It is worth noting that we divide the conception process into three types of behaviours: the conceptual labelling (C_1), constructive learning (C_2) and inductive learning (C_3) to demonstrate the methods via which the system builds a concept.

6.2.1. Behaviour records

In this section, we measure the above-mentioned behaviours of the system. The seven internal states are summarized in Table 5.

Table 4. *Symbols 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 – conception via constructive learning	Creating perceptual experience from memory 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	IAC neural network activation	Activating the perceptual experience structure (IAC) to get response
I_r	IAC neural network re-activation	Re-activating the perceptual experience structure (IAC) 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

Table 5. *Experiments with various design optimization scenarios and the agent's behaviours. A_c denotes activated experience. R_c shows the reactivated or initially validated experience. V_1 represents the validator state for A_c. H_s are hypotheses. V_2 describes the validator states for H_s. B_e is the abbreviation for the agent's behaviours. N_k means new knowledge learned. √ shows that the agent correctly predicts the situation and* X *shows the opposite.*

TASKS	A_C	V_1	R_C	H_S	V_2	B_E	N_K
Task 1	N/A	N/A	N/A	N/A	N/A	S, P, C_2	New Experience Ins-2
Task 2	Ins-1	√	N/A	N/A	N/A	S, P, I_a, C_1, V_d, W_a	Grounded Experience Ins-1
Task 3	Ins-1	√	N/A	N/A	N/A	S, P, I_a, C_1, V_d, W_a	Grounded Experience Ins-1
Task 4	Ins-2	N/A	N/A	N/A	N/A	S, P, I_a, C_1, V_d, W_a	Grounded Experience Ins-2
Task 5	N/A	N/A	N/A	N/A	N/A	S, P, C_2	New Experience Ins-3
Task 6	Ins-1	√	N/A	N/A	N/A	S, P, I_a, C_1, V_d, W_a, C_3	Grounded Experience Ins-1 and *New Concept 1*
Task 7	Ins-3	√	N/A	N/A	N/A	S, P, I_a, C_1, V_d, W_a, C_3	Grounded Experience Ins-3
Task 8	Ins-2	√	N/A	N/A	N/A	S, P, I_a, C_1, V_d, W_a, C_3	Grounded Experience Ins-2
Task 9	Ins-2	√	N/A	N/A	N/A	S, P, I_a, C_1, V_d, W_a, C_3	Grounded Experience Ins-2
Task 10	Ins-1 false memory	X	Ins-1,2,3 uncertain Memory	Quadratic Programming Reactivated Ins-1	X	S, P, I_a, C_1, V_d, I_r, C_1, H, I_r, C_2, C_3	New Experience Ins-4, and *New Concept 2*
Task 11	Ins-4 false memory	X	Ins-4 false memory	Quadratic Programming Reactivated Ins-1	√	S, P, I_a, C_1, V_d, I_r, C_1, H, I_r, W_a, C_3	Grounded Experience Ins-1
Task 12	Ins-4 false memory	X	Ins-4 false memory	Quadratic Programming Reactivated Ins-1	√	S, P, I_a, C_1, V_d, I_r, C_1, H, I_r, W_a, C_3	Grounded Experience Ins-1
Task 13	Ins-2	√	N/A	N/A	N/A	S, P, I_a, C_1, W_a, C_3	Grounded Experience Ins-2
Task 14	Ins-2	√	N/A	N/A	N/A	S, P, I_a, C_1, W_a, C_3	Grounded Experience Ins-2
Task 15	Ins-2	N/A	N/A	N/A	N/A	S, P, R_{ex}, W_a, C_3	Grounded Experience Ins-2

New Concept 1:
- OBJF_Type = Quadratic → Optimizer = Quad Programming;
- OBJF_Type = Linear → Optimizer = Lin-Programming;
- OBJF_Type = Nonlinear → Optimizer = Nonlinear-Programming;

New Concept 2:
- Provide_Hessian = false and OBJF_Type = Quadratic → Optimizer = Nonlinear-Programming
- Provide_Hessian = false and OBJF_Type = Linear → Optimizer = Lin-Programming
- Provide_Hessian = false and OBJF_Type = Nonlinear → Optimizer = Nonlin-Programming
- Provide_Hessian = true → Optimizer = Quad-Programming

6.2.2. Behaviour analysis

The behaviours recorded in this experiment are shown in Fig. 14 and Fig. 15. Fig. 14 shows changes of behaviours of the system from tasks 1 to 8, and Fig. 15 describes behaviours emerging from tasks 9 to 15.

We can identify a number of behaviour patterns from these two figures, which allows us to trace the role of the interaction in shaping the system's behaviours. Table 6 shows a number of behaviour patterns and the underlying causalities for their formations.

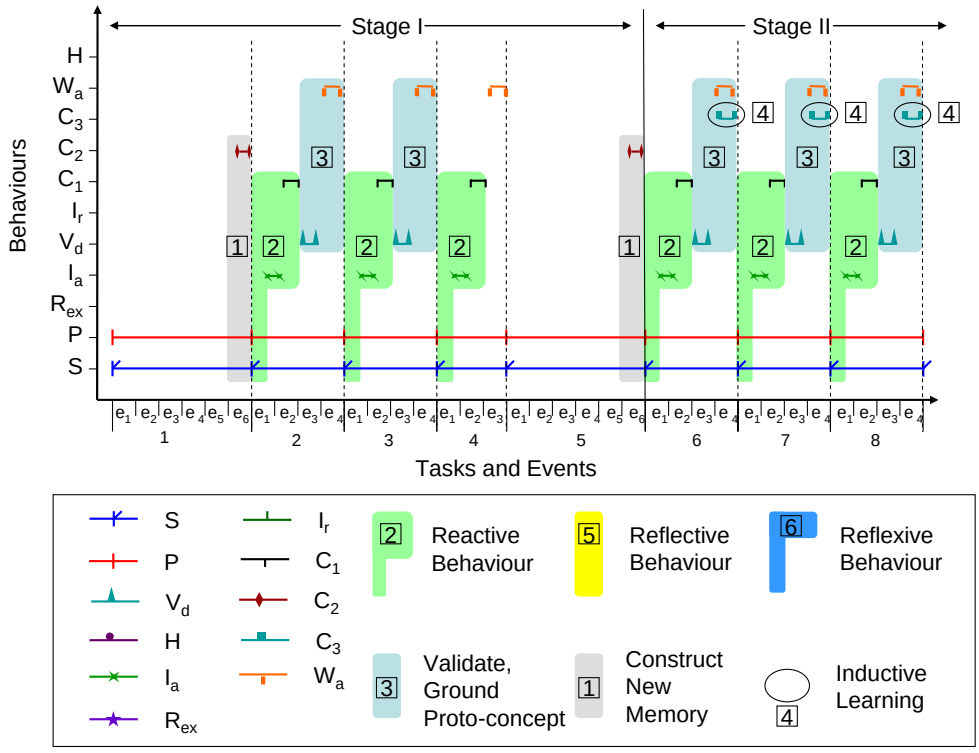

Fig. 14. Behaviour chart for the system during tasks 1–8.

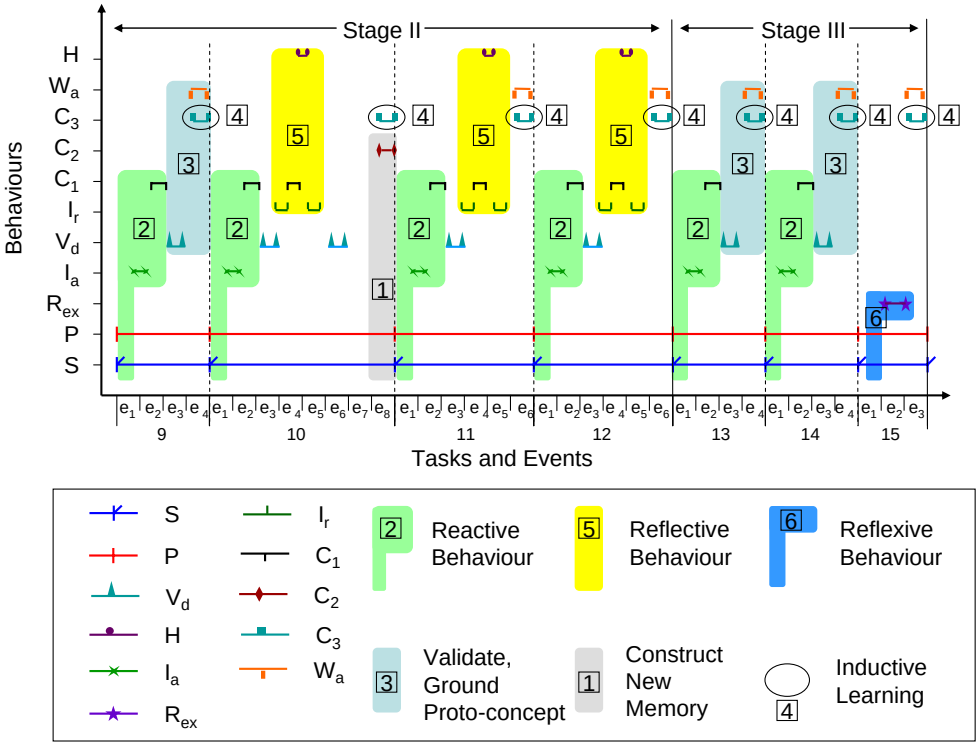

Fig. 15. Behaviour chart for the system during tasks 9–15.

Table 6. *Behaviour patterns, their formations and causalities.*

Pattern	Name	Descriptions	Causalities
1	Construct new memory	Activation sequence: S → P → C_2, Performing constructive learning. Behaviours marked with "1" in Fig. 14 and Fig. 15	The coordination of micro-interaction and macro-interaction when no previous experience is valid.
2	Reactive behaviour pattern	Activation sequence: S→ P → I_a → C_1, behaviours marked with "2" in Fig. 14 and Fig. 15	The coordination of micro-interaction and macro-interaction when previous experience is activated and constructed into a proto-concept.
3	Validate and ground the proto-concept	Activation sequence: V_d → W_a, behaviours marked with "3" in Fig. 14 and Fig. 15	The coordination of micro-interaction and macro-interaction when receiving affirmative feedbacks from macro-interaction.
4	Inductive learning	C_3, behaviours enclosed in ellipses and marked with "4" in Fig. 14 and Fig. 15	Micro-interaction that generalizes invariants from low-level experience.
5	Reflective behaviour pattern	Activation sequence: I_r → C_1 → H → I_r, behaviours marked with "5" in Fig. 15	The coordination of micro-interaction and macro-interaction. This involves reactivating the system's experience, using conceptual knowledge to deduce hypotheses and subsequently refocusing on (or creating) an existing (or new) proto-concept.
6	Reflexive behaviour pattern	Activation sequence: S → P → R_{ex}, behaviours marked with "6" in Fig. 15 (Task 15)	The coordination of micro-interaction and macro-interaction when the system has a very strong experience to an environmental stimulus.
Compound pattern 1	Pattern 2 → 3	React then ground (Tasks 2-3, 6-9, 13-14)	This happens when the system's perceptual level experience is useful.
Compound pattern 2	Pattern 2 → 5 → 1	React, reflect then construct new memory (Task 10)	This happens when the system's experience is not available.
Compound pattern 3	Pattern 2 → 5 → W_a	React, reflect, then reinforce the experience (Tasks 11-12)	This happens when the system's conceptual experience is useful in creating hypotheses.

We can further cluster the system's learning behaviour into three stages based on aggregations of these patterns: 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:

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

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 Fig. 16, Fig. 17 and Fig. 18. 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 Fig. 16.

.

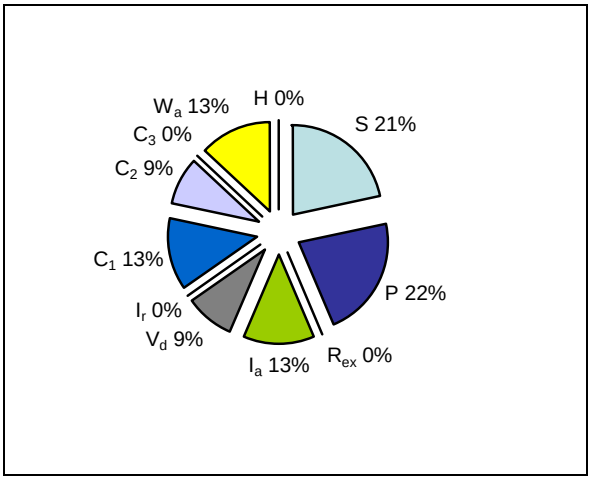

Fig. 16. Agent's behaviours in Stage I.

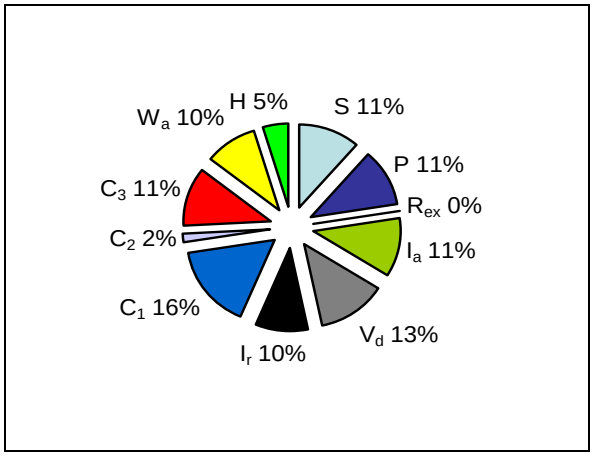

Fig. 17. Agent's behaviours in Stage II.

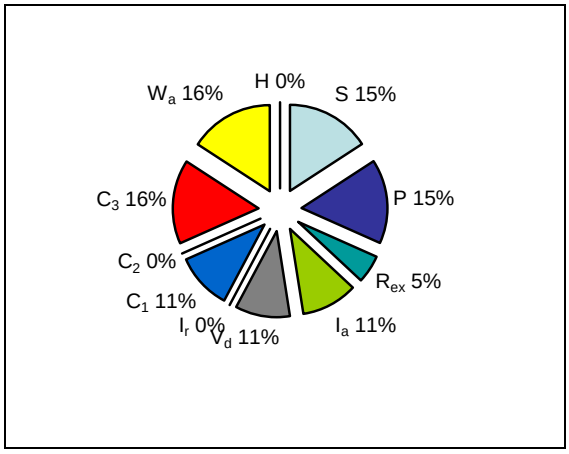

Fig. 18. Agent's behaviours in Stage III.

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. As illustrated in Fig. 17, 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.

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 Fig. 18.

A comparative study of these learning stages is presented in Table 7.

Table 7. The comparison of behaviours of the system in different stages; (A) represents the absolute value and (B) shows the percentage value.

		S	P	R_{EX}	I_A	V_D	I_R	C_1	C_2	C_3	W_A	H
Stage (A)	I	5	5	0	3	2	0	3	2	0	3	0
Stage (B)	I	21%	22%	0%	13%	9%	0%	13%	9%	0%	13%	0%
Stage (A)	II	7	7	0	7	8	6	10	1	7	6	3
Stage (B)	II	11%	11%	0%	11%	13%	10%	16%	2%	11%	10%	5%
Stage (A)	III	3	3	1	2	2	0	2	0	3	3	0
Stage (B)	III	15%	15%	5%	11%	11%	0%	11%	0%	16%	16%	0%

In Table 7, the light grey shade 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. 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. It can be observed that the system shows both a higher percentage (5% in Stage II, compared to 0% for the system in the other stages, dark grey shade in Table 7) and absolute value (3, compared to 0 for the system in the other stages, Table 7) of the hypothesizing processes in this stage. A new concept (Concept 2) is formed based on the agent's interaction with the environment.

The salient feature for Stage III (underlined numbers in Table 7) is that the system demonstrates a higher percentage of reflexive behaviour (5% against 0%) than those in the other two stages.

These results are also illustrated in Fig. 19, in which a comparative visual analysis can be performed to provide a cross-reference for these findings. Fig. 19 (a) shows that 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 at this stage, due to the lack of resources in generalization. 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. The stacked histogram in Fig. 19 (b) presents the distribution of each process in three stages.

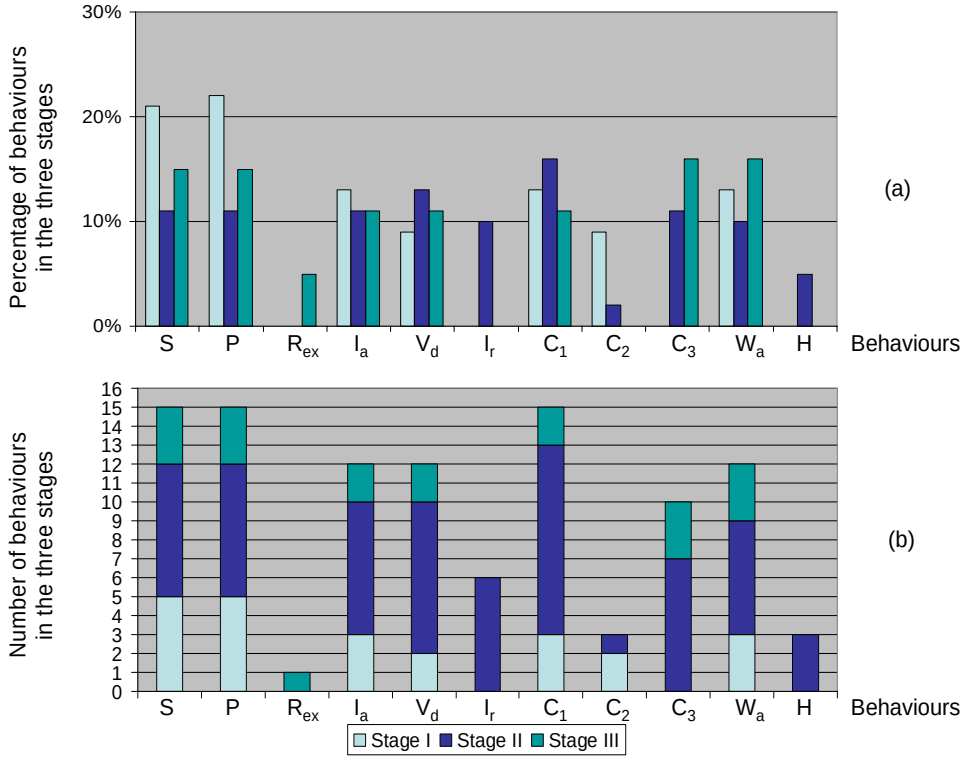

Fig. 19. A comparative study of the agent's behaviours and processes in various stages of learning: (a) shows percentages of behaviours in these three stages; (b) demonstrates absolute values of behaviours in the three stages.

These three stages can be explained by the internal structures created in the experiment. It is noted that 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 (Concept 1 in Table 5) enables the system to create hypotheses and therefore contributes to the system's reflective behaviour in 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.

In Experiment III, the agent exhibits similar behaviours to Experiment II. The conceptual knowledge (Concepts 1 and 2) gained from Experiment III are different from those from Experiment II, due to heterogeneous interaction schemes.

6.3. 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 scenarios that were used in Test II are 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.

Table 8 shows performances of this comparison experiment. The "0-1" loss function is applied to measure the outcomes of the prediction. It is more appropriate than using probability assessment in this test, because the ultimate application is merely a prediction of the outcome and the prediction is not subject to further processing (Witten & Frank, 2005). We use prediction success rate (P_s) to measure the overall performance of a system in this test:

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

Table 8. *Performances of three different systems.*

Design Tasks	Static System	Reactive System	Situated System
1	0	0	0
2	1	1	1
3	1	1	1
4	0	1	1
5	0	0	0
6	1	1	1
7	0	1	1
8	0	1	1
9	0	1	1
10	0	0	0
11	1	0	1
12	1	0	1
13	0	1	1
14	0	1	1
15	0	1	1

Table 9. *Confusion matrices for a situated system, P_s stands for Prediction Success Rate.*

Situated System		Predicted Class				
		Q	L	NL	Uncertain	Total
Actual Class	Q	5	0	0	0	5
	L	0	6	0	1	7
	NL	1	0	1	1	3
	Total	6	6	1	2	
P_s	(5+6+1)/15 = 0.8					

Table 9, Table 10 and Table 11 show confusion matrices for these three types of systems. Each matrix element shows the number of test examples for which the actual class is presented in the row and the predicted class is the column (Witten & Frank, 2005). For example, in row 2 of Table 9, the situated system predicts 7 instances of "L" (linear optimization problem), within which six instances are correctly predicted as "L" and one instance is an uncertain case.

Table 10. *Confusion matrices for a reactive system.*

Reactive System		Predicted Class				
		Q	L	NL	Uncertain	Total
Actual Class	Q	3	0	2	0	5
	L	0	6	0	1	7
	NL	1	0	1	1	3
	Total	4	6	3	2	
P_s	(3+6+1)/15 = 0.67					

Table 11. *Confusion matrices for a static system.*

Static System		Predicted Class				
		Q	L	NL	Uncertain	Total
Actual Class	Q	5	0	0	0	5
	L	0	0	0	7	7
	NL	0	0	0	3	3
	Total	5	0	0	10	
P_s	(5+0+0)/15 = 0.33					

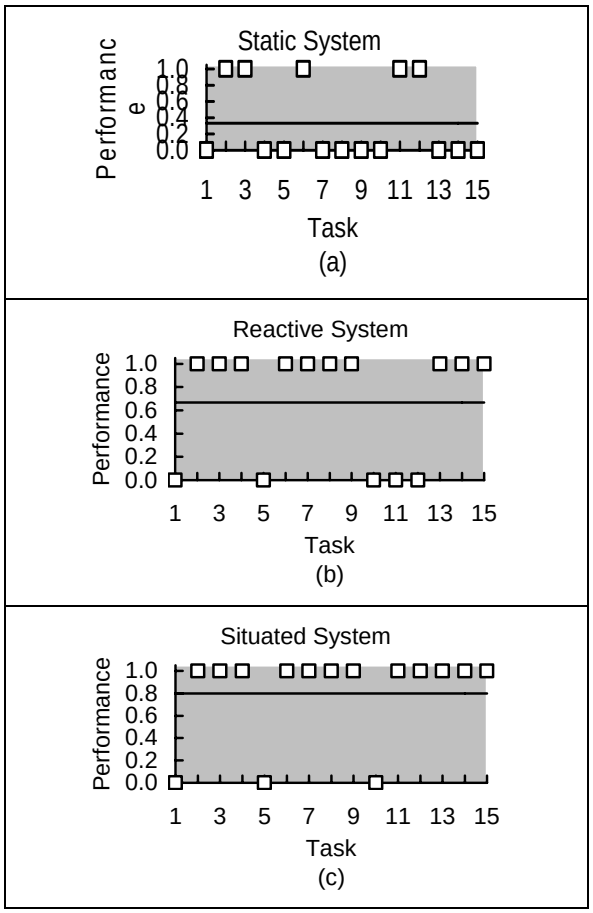

Fig. 20. (a) shows the prediction success rate for a static system; (b), (c) illustrate the prediction success rates for a reactive and a situated system.

The main diagonal elements (shaded cells in Table 9) show the correctly predicted classes.

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. As shown in the performance chart (Fig. 20), a situated system produces a prediction success rate of 0.8. 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.

7. CONCLUSION

Experiment 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. With regard to a static system based on pre-defined knowledge and a reactive agent which merely learns by the constructive learning, this situated agent-based design interaction tool performs better.

In summary, the proposed situated agent-based design interaction tool plays a potential role in supporting decision-making in a dynamic design process, where *a priori* knowledge is not adequate. The framework developed here may also lay foundations for future quests into adaptive and personalized design tools.

ACKNOWLEDGEMENTS

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.

REFERENCES

Balachandran, M.B. (1988). *A Model for Knowledge-Based Design Optimisation*. PhD Thesis. Sydney: University of Sydney.

Bartlett, F.C. (1932, reprinted in 1977). *Remembering: A Study in Experimental and Social Psychology*. Cambridge: Cambridge University Press.

Bisbey, P.R. & Trajkovski, G.P. (2005). *Rethinking Concept Formation for Cognitive Agents.* Working Paper, Towson University.

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

Chalmers, D.J. (1992). Subsymbolic computation and the Chinese room. In *The Symbolic and Connectionist Paradigms: Closing the Gap* (Dinsmore, J., Ed.), pp. 25–48. Hillsdale, NJ: Lawrence Erlbaum.

Clancey, W. (1995). A tutorial on situated learning. *Proc. Int. Conf. Computers and Education*, pp. 49-70. Charlottesville, VA: AACE.

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

Clancey, W. (1999). *Conceptual Coordination: How the Mind Orders Experience in Time*. New Jersey: Lawrence Erlbaum Associates.

Dewey, J. (1896, reprinted in 1981). The reflex arc concept in psychology. *Psychological Review*, *3*, 357–370.

Dorffner, G. & Prem, E. (1993). Connectionism, symbol grounding, and autonomous agents. *Proc. 5th Annual Meeting of the Cognitive Science Society,* pp. 144–148. Hillsdale, NJ: Lawrence Erlbaum.

Duffy, A.H.B., Persidis, A. & MacCallum, K.J. (1995). NODES: A numerical and object based modelling system for conceptual engineering design. *Knowledge-Based Systems*, *9(3)*, 183–206.

Fisher, D.H. & Pizzani, M. (1991). Computational models of concept learning. In *Concept Formation: Knowledge and Experience in Unsupervised Learning* (Fisher, D.H., Pazzani, M.J. & Langley, P., Eds), pp. 3–43. San Mateo, CA: Morgan Kaufmann.

Flemming, U.J. (1994). Case-based design in the SEED system. In *Knowledge-Based Computer-Aided Architectural Design* (Carrara, G., Kalay, Y.E., Eds), pp. 69–91. Amsterdam: Elsevier Science.

Gero, J.S. (1996). Design tools that learn: A possible CAD future. In *Information Processing in Civil and Structural Design* (Kumar, B., Ed.), pp. 17–22. Edinburgh: Civil-Comp Press.

Gero, J.S. (1998a). Conceptual designing as a sequence of situated acts. In *Artificial Intelligence in Structural Engineering* (Smith, I., Ed.), pp. 165–177. Berlin: Springer.

Gero, J.S. (1998b). Towards a model of designing which includes its situatedness. In *Universal Design Theory* (Grabowski, H., Rude, S. & Grein, G., Eds), pp. 47–56. Aachen: Shaker Verlag.

Gero, J.S. (1999). Recent design science research: Constructive memory in design thinking. *Architectural Science Review*, *42*, 3-5.

Gero, J.S. (2003). Design tools as situated agents that adapt to their use. *Proc. 21st Int. eCAADe Conf.*, pp. 177–180. Austria: Graz University of Technology.

Gero, J.S. & Fujii, H. (2000). A computational framework for concept formation in a situated design agent. *Knowledge-Based Systems*, *13(6)*, 361–368.

Gero, J.S. & Smith, G.J. (2006). A computational framework for concept formation for a situated design

agent, Part B: Constructive memory. Working Paper. Key Centre of Design Computing and Cognition, University of Sydney.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, *42*, 335–346.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D. & Rommelse, K. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. *Proc. Fourteenth Conf. Uncertainty in Artificial Intelligence*, pp. 256–265. Madison, WI: Morgan Kaufmann.

Huhns, M. & Acosta, R. (1992). Argo: An analogical reasoning system for solving design problems. In *Artificial Intelligence in Engineering Design* (Tong, C., Sriram, D., Eds), pp. 105–144. San Diego, CA: Academic Press.

Kalay, Y.E. (1999). The future of CAAD: From computer-aided design to computer-aided collaboration. *Proc. Eighth Int. Conf. Computer-Aided Architectural Design Futures,* pp. 14–39.

Lave, J. & Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. Cambridge. University of Cambridge Press.

Lieberman, H. (Ed.) (2001). *Your Wish is My Command: Programming by Example*. San Francisco: Morgan Kaufmann.

Lieberman, H. & Selker, T. (2000). Out of context: Computer systems that adapt to, and learn from, context. *IBM Systems Journal*, *39(3&4)*, 617–632.

Liew, P. & Gero, J.S. (2002). An implementation model of constructive memory for a situated design agent. In *Agents in Design 2002* (Gero, J.S., Brazier, F., Eds), pp. 257–276. Australia: Key Centre of Design Computing and Cognition, University of Sydney.

Lindblom, J. & Ziemke, T. (2002). Social situatedness: Vygotsky and beyond. Proc. 2nd Int. Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, pp. 71–78. Edinburgh, Scotland.

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

Maher, M.L. & Gero, J.S. (2002). Agent models of 3D virtual worlds. *Proc. ACADIA 2002,* pp. 127–138. Pomona, CA: California State Polytechnic University.

McClelland, J.L. (1981). Retrieving general and specific information from stored knowledge of specifics. *Proc. Third Annual Meeting of the Cognitive Science Society,* pp. 170–172. Hillsdale, NJ: Erlbaum.

McClelland, J.L. (1995). Constructive memory and memory distortion: A parallel distributed processing approach. In *Memory Distortion: How Minds, Brains, and Societies Reconstruct the Past* (Schacter, D.L., Ed.), pp. 69–90. Cambridge, Massachusetts: Harvard University Press.

McLaughlin, S. & Gero, J.S. (1987). Acquiring expert knowledge from characterised designs. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *1(2)*, 73–87.

Medler, D.A. (1998). A brief history of connectionism. *Neural Computing Surveys*, *1(1)*, 61–101.

Mostow, J. (1989). Design by derivational analogy: Issues in the automated replay of design plans. *Artificial Intelligence*, *40,* 119–184.

Mostow, J., Barley, M. & Weinrich, T. (1992). Automated reuse of design plans in bogart. In *Artificial Intelligence in Engineering Design* (Tong, C., Sriram, D., Eds), Vol. 2, pp. 57–104. San Diego, CA: Academic Press.

Myers, B. (1998). A brief history of human computer interaction technology. *ACM Interactions*, *5(2)*, 44–54.

Peng, W. & Gero, J.S. (2006). Concept formation in a design optimisation tool. *Proc. Design & Decision Support Systems 2006,* pp. 293–308. Berlin: Springer-Verlag.

Persidis, A. & Duffy, A. (1991). Learning in engineering design. In *Intelligent CAD III* (Yoshikawa, H., Arbab, F. & Tomiyama, T., Eds), pp. 251–272. Amsterdam: Elsevier Science.

Radford, A.D. & Gero, J.S. (1988). *Design by Optimization in Architecture and Building*. Reinhold. NY. Van Nostrand.

Reffat, R. & Gero, J.S. (2000). Computational situated learning in design. In *Artificial Intelligence in Design '00* (Gero, J.S., Ed.), pp. 589–610. Dordrecht: Kluwer Academic Publishers.

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

Rosenstein, M.T. & Cohen, P.R. (1998). Concepts from time series. *Proc. Fifteenth National Conf. Artificial Intelligence,* pp. 739–745.

Rutherford, J.H. & Maver, T.W. (1994). Knowledge-based design support. In *Knowledge-Based Computer-Aided Architectural Design* (Carrara, G., Kalay, Y.E., Eds), pp. 243–267. Amsterdam: Elsevier Science.

Schon, D. (1983). *The Reflective Practitioner: How professionals think in action,* London: Basic Books.

Selker, T. (1994). COACH: A teaching agent that learns. *Communications of the ACM*, *37(7)*, 92–99.

Smith, G. & Gero, J.S. (2000). The autonomous, rational design agent. *Workshop on Situatedness in Design, Artificial Intelligence in Design '00,* pp. 19–23. Worcester, MA.

Suchman, L.A. (1987). *Plans and Situated Actions: The Problem of Human-machine Communication*. Cambridge, University of Cambridge Press.

Vygotsky, L.S. (1978). *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA. Harvard University Press. (Original work published in 1934.)

Vygotsky, L.S. (1986). *Thought and Language*. Cambridge, Mass: MIT Press.

Witten, I.H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. San Francisco, CA: Morgan Kaufmann.

Wooldridge, M.J. & Jennings, N.R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, *10(2)*, 115–152.

Ziemke, T. (1999). Rethinking grounding. In *Understanding Representations in the Cognitive Sciences* (Riegler, A., Peschl, M. & Stein, A., Eds), pp. 177–190. New York: Plenum Publisher.