

# Using a Constructive Interactive Activation and Competition Neural Network to Construct a Situated Agent's Experience

Wei Peng and John S. Gero

Key Centre of Design Computing and Cognition  
University of Sydney, NSW 2006, Australia  
{wpeng, john}@arch.usyd.edu.au  
<http://www.arch.usyd.edu.au/kcdc/>

**Abstract.** This paper presents an approach that uses a Constructive Interactive Activation and Competition (CIAC) neural network to model a situated agent's experience. It demonstrates an implemented situated agent and its learning mechanisms. Experiments add to the understanding of how the agent learns from its interactions with the environment. The agent can develop knowledge structures and their intentional descriptions (conceptual knowledge) specific to what it is confronted with – its experience. This research is presented within the design optimization domain.

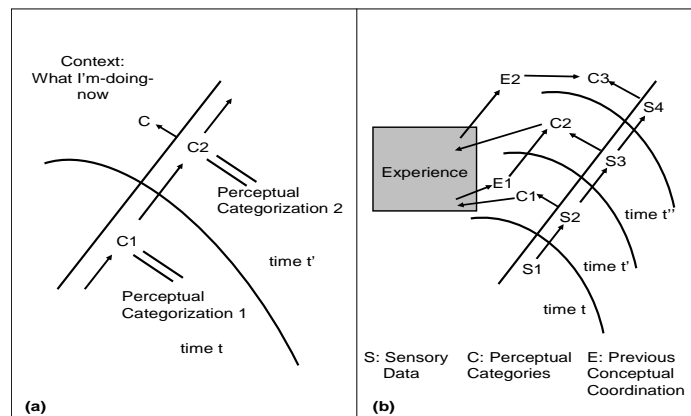
## 1 Introduction

Experience is defined as the accumulation of knowledge or skill that results from direct participation in events or activities [15]. Learning is a process whereby knowledge is created through the transformation of experience [6], [8]. Experience plays a key role in a learning process. Learning, also known as knowledge acquisition, has been widely explored by many artificial intelligence (AI) and machine learning researchers. A broad spectrum of approaches has been developed, including inductive learning methods, explanation-based learning approaches and connectionist algorithms. Another theory of learning emerged from cognitive science domain. "Situatedness" [1], [4] and "situated learning" [9], [13] emphasize the role of social interactions in learning. An agent's memory can be regarded as a learning process. The notion of "constructive memory" contradicts many views of knowledge as being unrelated to either its locus or application [5]. A constructive memory model [5], [10] provides a conceptual framework for us to utilize the concept of "situatedness" in a software agent. Learning takes place when a learner interacts with, or is stimulated by, an environment [6]. An agent relates its own experience and gives meanings to a situation [7]. These two theories are not incompatible, but complementary, with both addressing the same issue at different levels. An agent that is designed to be situated at a conceptual level can still be implemented using various machine learners. In this paper, we describe how to model a situated agent's experience using a Constructive Interactive Activation and Competition (CAIC) neural network.

This situated agent 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. Choosing a suitable optimizer becomes the bottleneck of a design optimization process. Designers rely on their experience to carry out this task. Such a manual process may result in a sub-optimal design solution and hence an inefficient design. Our objective is to construct a computational model which is able to capture the knowledge of using the design optimization tool, and as a consequence, can aid the tool's future use in supporting design. 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 task, the same tool constructs memories of a design situation and anticipates the tool's potential use. It can therefore offer helps to designers in their interactions in designing even before they require. Our approach is to use a situated agent that wraps around an existing design optimization tool. A user accesses a design tool via this wrapper, where a situated agent senses the events performed by that user and learns new concepts from the user's interactions with it.

## 2 Situated Agency

Software agents are intentional systems that work autonomously and interact with environments in selecting actions to achieve goals [14]. A situated agent is the software that is founded on the notion of "situatedness".



**Fig. 1.** (a) shows the conceptual knowledge as a higher order categorization of a sequence (after Fig. 1.6 of Clancey [2]). (b) illustrates a situated learning scenario

Situatedness involves both the context and the observer's experiences and the interactions between them. Situatedness [1] holds that "where you are when you do

what you do matters” [4]. Conceptual knowledge can be learned by taking account of how an agent orders its experience in time, which is referred as conceptual coordination [2], Fig. 1.

Conceptual knowledge is a function of previously organized perceptual categories and what subsequently occurs, Fig. 1(a). It is generally formed by holding active a categorization that previously occurred (C1) and relating it to an active categorization C2 [2]. Fig. 1(b) illustrates a scenario of 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’s experience to obtain similar organizations. The agent’s experiential response (E1) represents the agent’s hypotheses about what would happen later in the environment. 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 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 (C2) with the latest environmental changes.

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. 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:

$$S(t) = \{ \dots \text{“click on objective function text field”}, \text{key stroke of “x”}, \text{“(”}, \text{“1”}, \text{“)”}, \text{“+”}, \text{“x”}, \text{“(”}, \text{“2”}, \text{“)”} \dots \} \quad (1)$$

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:

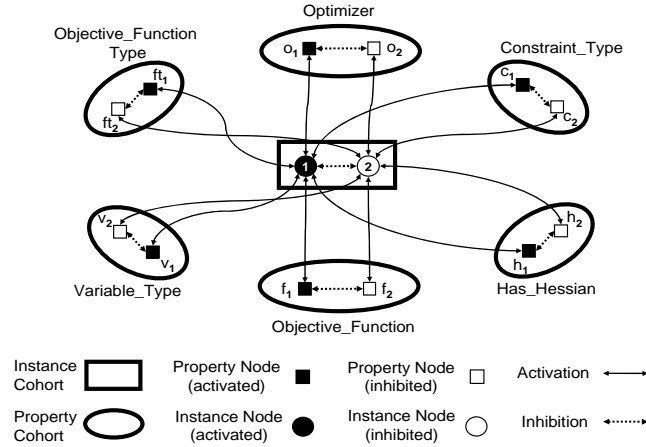
$$P(t) = (\text{Object}, \text{Property}, \text{Values of properties}) \quad (2)$$

For example, a perceptual data  $P_1$  can be described as (Objective Function Object, Objective\_Function, “x(1)+x(2”).

The cue\_Maker generates cues that can be used to activate the agent’s experience. The conceptor categorizes the agent experience to form concepts. Concepts attach meanings to percepts. The hypothesizer generates hypotheses from the learned concepts. This is where reinterpretation takes place in allowing the agent to learn in a “trial and error” manner. The validator pulls information from the environment and examines whether the environmental changes are consistent with the agent’s responses. An agent needs to validate its hypotheses in interactions to locate a suitable concept for the current situation. An effector is the unit via which the agent brings changes to environments through its actions.

The agent’s experience is structured as a Constructive Interactive Activation and Competition (CIAC) neural network, in which we extend a basic IAC network [11] to accommodate the concept learning process, Fig. 2. An IAC consists of two basic nodes: instance node and property node. 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 [12]. Property nodes are grouped into cohorts of mutually exclusive values [12]. Each property node represents the perceptual level experience which is processed from sensory data.



**Fig. 2.** A CIAC neural network as a representation of the agent’s experience; Property nodes are labeled by their value, e.g.  $f_1$  represents a property node in the Objective\_Function cohort with objective function value  $f_1$

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 [12]. In Fig. 2, the shaded instance node (Ins-1) and related shaded property nodes presents a context addressable memory cued from an environment stimulus, e.g. [Objective\_Function,  $f_1$ ]. Such a response is a dynamic construction in the sense that when environment stimuli change, the agent develops adapted knowledge. This organized experience grounds by weight adaptation and constructive learning.

### 3 Constructing a Situated Agent’s Experience

In this section, we discuss how to construct a situated agent’s experience with a CIAC neural net. Table 1 illustrates the formulas that are used to compute network input value ( $N_c$ ) and activation value ( $A_c$ ) for each nodes during activation and competition phrase. Table 1 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 [12].

**Table 1.** Major formulas applied in the CIAC neural net

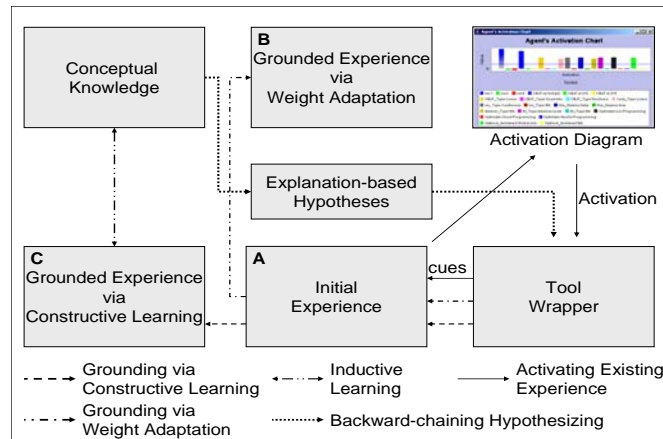
Items	Fomulas
<p>Network Input Values (<math>N_e</math>):</p> <p>-- the activations of each of the neurons can be regarded as the “degree of belief” in that hypothesis</p>	$N_e = \mu E + \sum_{i=1}^n W_i A_i$ <p><math>\mu</math>: excitatory gain for initial network stimulus, set to 4.0  <math>E</math>: initial network stimulus, default 1.0  <math>W_i</math>: inbound weights for a neuron  <math>A_i</math>: activation value for each inbound weight  <math>n</math>: number of neurons in a network</p>
<p>Activation Values (<math>A_c</math>):</p> <p>-- the tendency for units connected by negative weights to turn each other off</p> <p>-- a unit that begins with a small advantage “wins” the competition and becomes active at the expense of the other units in the pool in the end [3].</p>	<p><i>If</i> <math>N_e &gt; 0</math></p> $A_c = A_{c-1} + \ell [(A_{\max} - A_{c-1})N_e - \varphi(A_{c-1} - R)]$ <p><i>else</i></p> $A_c = A_{c-1} + \ell [(A_{c-1} - A_{\min})N_e - \varphi(A_{c-1} - R)]$ <p><math>A_c</math>: the activation value for each neuron at current cycle  <math>A_{c-1}</math>: the activation value for that node at previous cycle  <math>N_e</math>: the net input for each node  <math>A_{\max}</math>: the maximum permitted activation value, set to 1.0  <math>A_{\min}</math>: the minimum permitted activation value, set to -0.2  <math>R</math>: the initial resting activation value, default -0.1  <math>\varphi</math>: the decay factor, default 1.0  <math>\ell</math>: the learning rate, default 0.1;</p>
<p>Weight Adaptation (<math>W_n</math>):</p> <p>-- the process that verifies the usefulness of a related experience in current situation.</p> <p>-- similar to a Hebbian-like learning mechanism</p>	<p><i>If</i> <math>W_o &gt; 0</math></p> $W_n = W_o + \ell (W_{\max} - W_o) A_i A_j - \delta W_o$ <p><i>else</i></p> $W_n = W_o$ <p><math>W_o</math>: the weight value before weight-adaptation  <math>W_n</math>: the weight value after weight-adaptation  <math>\ell</math>: the learning rate, default 0.1  <math>W_{\max}</math>: the maximum permitted weight value, set to 1  <math>A_i, A_j</math>: activation values for neuron <math>i</math> and <math>j</math>  <math>\delta</math>: the weight decay factor, set to 0.1</p>

The pseudo code below (Table 2) presents procedures for constructing a situated agent’s experience with a CIAC neural network. It shows the relationships between functions in the experience package. Functions related to other packages are denoted by capitals of their package names. For example, `Perceptor.generateCue()` depicts the method belongs to the Perceptor package.

The implemented prototype system is illustrated in Fig. 3. 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. Activation diagrams output the neurons winning at the equilibrium state, which represent the activated memory.

**Table 2.** Constructing a situated agent's experience with a CIAC neural net

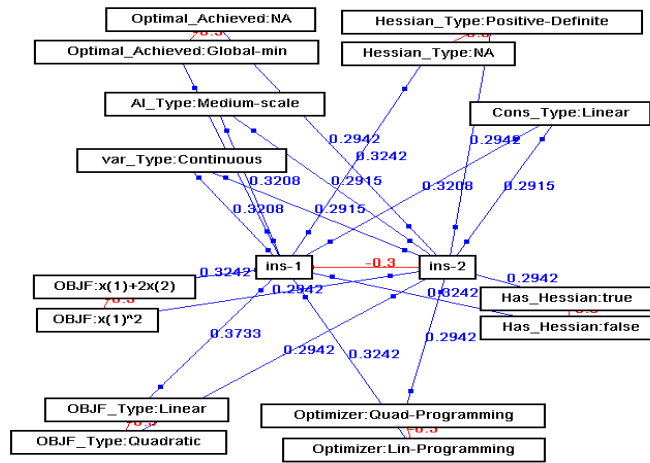
1.	Initial experience $\leftarrow$ CIAC net build from a matrix file
2.	Sensory data $\leftarrow$ GUISensor.catcher() //captures a user's actions
3.	Percepts $\leftarrow$ Perceptor.modalitySwitch() //data tranform
4.	Cues $\leftarrow$ Perceptor.generateCue() //create memory cues from percepts
5.	Activation: set activation values of the cue nodes to 1.0000
6.	<i>Cycling</i> : while CIAC net not isEquilibrium() //check equilibrium states
7.	<i>for all</i> nodes in the CIAC neural net, compute $N_e$ for each nodes
8.	<i>for all</i> nodes in the CIAC neural net, update $A_c$ for each nodes
	<i>End Cycling</i> //reach equilibrium states
9.	Output activated nodes //output nodes with activation value > threshold
10.	Percepts $\leftarrow$ Validator.pull() //check environment changes, transfer data
11.	result $\leftarrow$ Validator.isValid() //check the usefulness of the activated experience
12.	<i>If</i> result = true
13.	Grounding via weight adaptation() // Hebbian-like learning
	<i>Else</i> {
14.	Percepts $\leftarrow$ Validator.pull() //update and pull more environment changes
15.	hypotheses $\leftarrow$ Hypothesizer.hypothesize(Percepts, Concepts, Target)
16.	<i>output</i> hypotheses
17.	result $\leftarrow$ Validator.isValid() //check hypotheses
18.	<i>If</i> result = true
19.	Grounding() //weight adaptation via Hebbian-like learning
	<i>Else</i> { Constructive_learning() //incorporate new experience
20.	Concepts $\leftarrow$ Concepter.learnInductiveConcepts() } }
21.	New experience



**Fig. 3.** Constructing a situated agent's experience

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

process initiates a validation function which matches the initially constructed concepts with environmental changes. The weight adaptation function 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 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 are obtained from applying an inductive learner to the agent’s experience. These conceptual knowledge serve as domain theories, from which the agent creates hypotheses. A typical grounded experience is illustrated as below.



**Fig. 4.** A typical experience grounded from an initial experience which has 1 instance node connected to several property nodes with weights “0.3000”. Property nodes are described as property and value pairs, e.g. “OBJF\_Type: Linear” represents a property node with linear objective function type

## 4 Agent Behaviours and Experiments

This system provides a basis for us to explore the behaviours of a situated agent in various situations. We examine how the agent learns new concepts, in terms of developing knowledge structures and their intentional generalizations in its interactions with the environment. The following five internal states and their changes can be used to study how an agent constructs concepts:

1. The knowledge structure which is a Constructive Interactive Activation and Competition (CIAC) neural network composed of instance nodes connected to a number of property (or feature) nodes;
2. The expectation about environmental changes are generated by the agent experiential responses to environmental cues (shown in the activation diagram in Fig. 3);

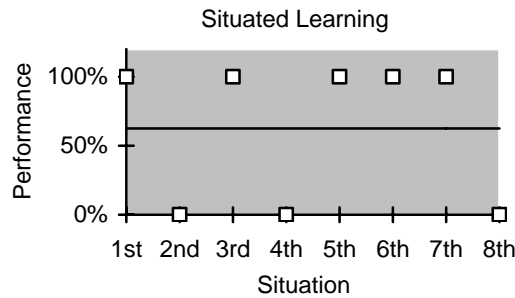
3. The validator states show whether an agent's expectation is consistent with the environment changes;
4. Hypotheses depict the agent's reinterpretation about its failures in creating a valid expectation;
5. Concepts are the agent's high-level experiences which are domain theories an agent uses to classify and explain its observations.

**Table 3.** Experiments with various design optimization scenarios and the agent's behaviours.  $A_c$  denotes activated 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. QP stands for a quadratic programming optimizer. NLP is a nonlinear programming optimizer.  $\checkmark$  shows that the agent correctly predicts the situation and  $\times$  shows the otherwise. "Ins-1" stands for design experience instance 1 and "OBJF" is the abbreviation for objective function. "Cons" represents constraints and "HF" is for Hessian function.

Design Scenarios	$A_c$	$V_1$	$H_s$	$V_2$	$B_e$	$N_k$
Identical OBJF to ins-1, Linear Cons	Ins-1	$\checkmark$	N/A	N/A	Grounds Ins-1	Grounded Ins-1
Linear OBJF, No Cons	Ins-1	$\times$	N/A	N/A	Constructs Ins-2	New Experience Ins-2
Linear OBJF, Linear Cons	Ins-1	$\checkmark$	N/A	N/A	Grounds Ins-1 Constructs Ins-3	Grounded Ins-1 and New Ins-3
Quadratic OBJF and No Cons	None	$\times$	N/A	N/A	Constructs Ins-4	New Experience Ins-4
Quadratic OBJF and No Cons	Ins-4	$\checkmark$	N/A	N/A	Grounds Ins-4 Constructs Ins-5	Grounded Ins-4 and New Ins-5
Quadratic OBJF and No Cons	Ins-4,5	$\checkmark$	N/A	N/A	Grounds Ins-4,5 Constructs Ins-6 Inductive Learning	Grounded Ins-4, 5 and New Ins-6; <i>New Concepts 1,2</i>
Quadratic OBJF, Linear Cons	Ins-4,5,6	$\times$	Is a QP	$\checkmark$	Hypothesize, Grounds Ins-4,5,6 Constructs Ins-7	Grounded Ins-4, 5, 6 and New Ins-7;
Quadratic OBJF, Linear Cons, no HF	Ins-4,5,6,7	$\times$	Is a QP	$\times$	Hypothesize, Constructs Ins-8	New Ins-8, <i>New Concepts 3,4</i>

- New Concept 1: OBJF\_Type = Quadratic → Optimizer = Quad-Programming;
- New Concept 2: OBJF\_Type = Linear → Optimizer = Lin-Programming;
- New Concept 3: OBJF\_Type = Quadratic and Provide\_Hessian = false → Optimizer = Nonlin-Programming;
- New Concept 4: OBJF\_Type = Quadratic and Provide\_Hessian = true → Optimizer = Quad-Programming.

The initial experience of the agent holds one instance of a design optimization scenario using a linear programming algorithm. An experiment has been carried out to study the learning behaviours of a situated agent in heterogeneous design optimization scenarios over time. The performance is defined as the correctness of the system's response to an environment cue, which predicts hence, assists the applied design task. The "0-1" loss function is applied to measure the outcomes of the prediction. The results are illustrated in Table 3. From these results of this experiment, we can see that the agent develops its experience through reorganizing existing experience or constructing a new design experience from its interaction with the environment. It adapts to the environment with the learned knowledge, ranging from detailed design instances to generalizations of these low-level experiences, i.e., new concept 1-4 in Table 3. As shown in Fig. 5, even at the early stage of its learning, the agent achieves a performance of 62.5% in recognizing design optimization problems.



**Fig. 5.** Performance of the initial stage of experience learning in a situated agent. The *square dot* shows the performance of the agent in recognizing new situations in the experiment. The *dark black line* represents the mean squared estimation of the agent's performance (62.5%)

We conjecture one of the reasons for this is the content addressable ability of an CIAC neural net which can generalize across exemplars and provide plausible default values for unknown variables [3]. A situated agent that can inductively learn new concepts and subsequently deduce explanations for environment changes also adds a new level of learning to this CIAC neural net.

## 5 Conclusions

In this paper, we have described an approach that applies a Constructive Interactive Activation and Competition (CIAC) neural network to model a situated agent's experience. We demonstrate an implemented situated agent that uses a constructive

memory model to learn new concepts from its interaction with the environment. From the results obtained from the experiment, we can conclude that the agent develops its knowledge structures and behaviours specific to what it is confronted with – its experience. Based on the conceptual knowledge learned, the agent can further improve its learning behaviour. As a result, designers can integrate their expertise with the knowledge learned from the agent to develop design solutions. Such a system plays a potential role in enhancing the design optimization efficiency.

**Acknowledgments.** 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

1. Clancey, W.: *Situated Cognition*. Cambridge University Press, Cambridge (1997)
2. Clancey, W.: *Conceptual Coordination: How the Mind Orders Experience in Time*. Lawrence Erlbaum Associates, New Jersey (1999)
3. Dennis, S.: *The Interactive Activation and Competition Network: How Neural Networks Process Information*. <http://www.itee.uq.edu.au/~cogs2010/cmc/chapters/IAC/> (1998)
4. Gero, J.S.: *Towards a Model of Designing which includes its Situatedness*. In: Grabowski, Rude and Grein (eds.): *Universal Design Theory*. Shaker Verlag, Aachen (1998) 47-56
5. Gero, J.S.: *Constructive Memory in Design Thinking*. In: Goldschmidt and Porter (eds.): *Design Thinking Research Symposium: Design Representation*, MIT, Cambridge (1999) 29-35
6. Hansen, R.E.: *The Role of Experience Learning: Giving Meaning and Authenticity to the Learning Process*. *Journal of Technology Education*.11 (2000) 23-32
7. Jarvis, P.: *Meaningful and Meaningless Experience: Towards an Analysis of Learning from Life*. *Adult Education Quarterly*. 37 (1987) 164-172
8. Kolb, D.A.: *Experiential Learning: Experience as the Source of Learning and Development*. Prentice Hall, Englewood Cliffs (1984)
9. Lave, J. and Wenger, E.: *Situated Learning: Legitimate Peripheral Participation*. University of Cambridge Press, Cambridge (1991)
10. Liew, P.: *A Constructive Memory System for Situated Design Agents*. PhD Thesis, Key Centre of Design Computing and Cognition, University of Sydney, Sydney, Australia (2004)
11. 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*, Erlbaum, Hillsdale, NJ (1981) 170-172
12. Medler, D.A.: *A Brief History of Connectionism*. *Neural Computing Surveys*. 1 (1998) 61-101
13. Reffat, R. and Gero, J.S.: *Computational Situated Learning in Design*. In: Gero, J. S. (eds.): *Artificial Intelligence in Design'00*. Kluwer Academic Publishers, Dordrecht (2000) 589-610
14. Wooldridge, M. and Jennings, N.R.: *Intelligent Agents: Theory and Practice*. *Knowledge Engineering Review*. 10 (1995) 115–152
15. Wordnet. <http://wordnet.princeton.edu.au/>