COMPUTATIONAL SITUATED LEARNING IN DESIGN

Application to Architectural Shape Semantics

RABEE M. REFFAT AND JOHN S. GERO
Key Centre of Design Computing and Cognition
The University of Sydney
Email {rabee, john@arch.usyd.edu.au}

Abstract. This paper presents the development of a computational process model of **S**ituated **L**earning **i**n **De**sign (**SLiDe**). Situated learning is based on the notion that knowledge is more useful when it is learned in relation to its immediate and active context, ie its situation, and less useful when it is learned out of context. The usefulness of design knowledge is in its operational significance based upon where it was used and applied. SLiDe elucidates how design knowledge is learned in relation to its situation, how design situations are constructed and altered over time in response to changes taking place in the design environment. SLiDe is implemented within the domain of architectural shapes in the form of floor plans to capture the situatedness of shape semantics. SLiDe utilises an incremental learning clustering mechanism not affected by the concept drift that makes it capable of constructing various situational categories and constantly modifying them over time. The paper concludes with a discussion of the potential benefits of using SLiDe during the conceptual stages of designing.

1. Introduction

In this paper, "designing" is used to refer to the activities carried out by designers during the design process and "design" refers to the product as design artefacts. Designing is an ill-defined problem whereby designers' actions are not based on performing a complete plan or a program that is given at the beginning of designing or even a priori. Furthermore, the result of designing is not based on actions independent of what is being designed or independent of when, where and how it has been designed. Design actions are continuously affected by such changes. Designers' actions are situation dependent, ie situated, such that designers work

interactively with the design environment within the specific conditions of the situation (Gedenryd, 1998). During the design process the relationships between individual design elements change over time based upon changes of designers' actions through their interactions with the design environment that is continuously affected by such changes.

The central idea of situated cognition is that in order to function efficiently, the brain needs not only the body but also the surrounding environment. Situatedness in action (Clancey, 1997) holds that "where you are when you do what you do matters". The implication of this view is that actions are interrelated to their locus and application. Much of artificial intelligence had been based on a static world. However, in designing the world is fluid whereby design elements are generated and emerged via dynamic and situated activities instead of fixed plans.

In this work we view designing as a situated and dynamic activity. Thus, situatedness in designing is concerned with locating design knowledge in its environment or its active context so that the actions taken are a function of both the situation and the way the situation is constructed or interpreted. The important characteristic of this view is the dynamic change in design context while designing whereby the whole design context constitutes the design environment and for the specific knowledge in focus the active and immediate part of the context, the situation, plays a significant role. The work presented in this paper aims to contribute to the development of the next generation of CAD systems.

Designing acts and knowledge are composed such that subsequent experiences categorise of what was experienced before. The categorisation of design knowledge and actions is based on regularities in observable phenomena. If there is no regularity then the phenomenon appears to be fortuitous. The elicitation of these regularities is a form of learning. The view of situated learning considered in this paper is to find regularities of interrelationships among design knowledge based upon where they were used to situate design knowledge within their active contexts. Hence, each certain piece of knowledge will carry with it notions of its applicability conditions derived from the situation. Thus, such regularities form the grounding to situate that knowledge. These applicability conditions are then modified over time as the regularities of the interrelationships are either changed or found to appear frequently in the design environment.

In this paper, we present the development of a computational process model of situated learning in design that associates design knowledge to the situation within which it was used and applied. Section 2 introduces the definition of the "situation" as adopted in this work and how the situation is constructed. An overview of situated learning about

architectural shape semantics is presented in Section 3. The process model's framework and results within the domain of architectural shape semantics are discussed in Section 4. The potential benefits of using such results are addressed in Section 5.

2. Situations and constructing the situatedness of knowledge

A dictionary definition (Merriam-Webster's) of a "situation" is the way in which something is placed in relation to its surroundings at a certain moment. The interpretation of this definition in design terms is the relevant parts of the environment in relation to a specific aim or focus. These relevant parts are those that interact with the design process and as a consequence have an effect on it.

We borrow from Barwise and Perry (1983) the idea that a situation is composed of a collection of entities. Each entity may have a set of properties associated with it. Furthermore, there is some specified relation of the entities to one another in the situation supporting the understanding of the situation in terms of how the entities relate to one another. The entities of a situation can be any meaningful characteristic or abstract idea. The relations of a situation are the meaningful associations among entities that provide structure to the situation. These include, but are not limited to, spatial and temporal relations. The spatial temporal location provides the framework for the situation.

Following Barwise and Perry (1983), we make a distinction between at least two different types of situations: static and dynamic situations. A static situation is a state of affairs in which the major components of the situation do not change. It involves the same collection of entities, in the same relation to one another and continuous spatial temporal location. In contrast, a dynamic situation is a course of events and composed of a series of related event frames that are linked through the commonality among them. To make this distinction clearer, consider the following. A mail carrier making a phone call in a static situation in which the entities, their properties and their interrelationships remain the same through the situation. In contrast, the situation of the mail carrier delivering the mail in a route is a dynamic situation where different entities such as different houses, streets, scenes, obstacles, etc. and different relationships are involved (Radvansky and Zacks, 1997).

The following is a simple analogy for constructing and learning situations. Imagine that you are reading a particularly engaging whodunnit where you are trying to solve the mystery before the author hands the solution to you at the end of the book. One of the basic elements of this task is to try to construct the circumstances under which the murder took

place. This may include the location of important objects at the crime scene, the location of other people at the time of the murder, the relations of different characters to one another and the victim and other pieces of information. To make the task more difficult, those aspects of the crime scene that you are allowed to learn about are revealed at different points of time and usually not in order of their importance. To be successful, you must integrate this information to construct the situation in which the murder took place (Radvansky and Zacks, 1997).

In this work, the role of situated learning in design is to associate knowledge with its situation by capturing the regularities of relevant relationships among shape semantics across different observations from the representations. For instance, a table in an office space environment and in a dinning space environment whereby the relationships between this table and other entities in an office space environment are regularities across observations recognised in that environment. On the other hand, there is another type of relationships between that table, however it may be the same object, and other different entities in a dining space environment. These regularities between relations are not natural laws neither causal effects, but rather an automatic consequence of differentiating the relations in the first place. It is these relations that allow the situated knowledge about a desk table and a dining table to constructed in relation to situations from the environment where a desk table or a dining table was used. Within the example, we may make a clear distinction between what we mean by context and situation. In an office space environment there are many objects, ie entities, that surround a desk table, such as drawers, file cabinets, computers, desk lamp, walls, windows, etc. These objects constitute the whole context of a desk table. In an open office space environment, wall and windows are not salient features of the desk table's surroundings, however the table is still to be recognised as desk table. So, the situation is only that active, immediate and specific part of the context to the focus

One of the main characteristics of designing is its dynamic nature. During the process of designing, both knowledge and situations are not treated as static, but are invariably subject to change. This is due to the change in the design environment, which involves searching for knowledge, structuring and interpretation. Most important, it involves the construction of knowledge-relationships during this cyclic process and joining these knowledge-relationships with the previous ones, defining new knowledge-relationships structures, which may lead to modify previous situations or create new ones. Once a situation has been constructed, it can be updated to include new knowledge that is relevant to the situation. Thus updating includes adding new knowledge that was not previously available or creating new situations based on the new

observed knowledge from the environment. The situations represent the applicability conditions of design knowledge. Design knowledge becomes situated when its applicability conditions are learned. So what is to be learned are the applicability conditions of design knowledge.

3. Situated learning vs. context free learning in design

All learning occurs in context (Balsam, 1985) and many practical learning applications necessitate the use of context in learning (Matwin and Kubat, 1996). Activity theory (Nardi, 1996) proposed a very specific notion of the context that makes it related to some extent to the definition of the situation as adapted in this work. What takes place in an activity composed of objects, actions and operation is the context. Context is constituted through the enactment of an activity. Relatively similar in AI terms context means the general conditions or circumstances in which an event and action takes place (Akman and Surav, 1995). Situated learning is not treating knowledge as static, but rather view categorisation as dynamic processes. What constitutes knowledge and appropriate criteria must be adjusted with every situation, otherwise there would be no situation to speak of. In effect, situated learning calls for the kind of metacognitive approach but the emphasis is not in learning a fixed strategy or way of organising the world, but understanding the context in which people are operating and focus down to design genuinely useful learning tools in the context of use. Situated learning theory reveals the limitations of computer-human interaction analysis based on descriptive and stored view of knowledge. Such assumptions led to use computers for automation and to replace people and teachers by machines; storing knowledge in the computers and deliver expert systems to workers and intelligent tutoring systems to students (Clancey, 1995).

In this work we are not intending to create a tool that simulates or models human perception, cognition and behaviour abilities but rather simply aim to make computers more useful to designers, through learning the situatedness of design knowledge. The main difference between situated learning and other forms of machine learning in design approaches is that, instead of a context free generalisation from cases or examples during knowledge acquisition process, knowledge is generalised with respect to the situation within which it was used. Machine learning paradigms are used within a shared view of the role of machine learning in design: namely that of "learning to perform existing tasks better using available tools", where the tools themselves are unchanged and learned

knowledge is either unrelated to its locus or application (Gero, 1996; Gero, 1999). As for the knowledge being unrelated to its locus or application makes the learned knowledge context free and universally applicable. When each of these designers tackles a similar design task it would be useful if the same tools now had knowledge about what it has learned and its relation to the situation within which it was learned. The effect of this would be tools which are increasingly useful to the designer (Gero, 1996). In order for this to occur and to guide the use of knowledge, tools would have to learn along with the knowledge concepts of its situatedness.

Situatedness of learning in design opens a different perspective of designing and learning that has not been adequately explored. Based upon the aforementioned terms of situatedness the vast majority of learning systems in design deal with the environment independently from the situational conditions, ie context free systems. The situatedness includes a set of concepts such as the dynamic nature of design knowledge and the interrelationships between knowledge and its locus and application. There are some machine learning design systems that are related in ethos to one of these concepts that is the dynamic nature of learning design knowledge. Learning incrementally and detecting change in an environment is important to knowing when it is time to learn. ECOBWEB (Reich and Fenves, 1992), is a systems that learns synthesis knowledge and have the ability to track a changing domain. Some other learning mechanisms that are responsive to changes in the environment and consider the issue of concept drift are STAGGER (Schlimmer and Granger, 1986), COBBIT (Kilander and Jansson, 1993) and Widmer (Widmer and Kubat, 1996). Some of ther learning systems in design that are somehow related in ethos to situated learning are BRIDGER (Reich, 1993) and PERSPECT (Duffy and Kerr, 1993; Kerr, 1993). BRIDGER uses an incremental learning scheme for the creation of hierarchical classification tress and ECOBWEB is a major component of it. It partially implements the constructive induction mechanism (Reich, 1991) for the incremental concept formation. In constructive induction, collection of design examples are used by a learning system to produce sequence of collection of design rules, each representing different semantics. Within the situated, design knowledge captured is these rules are unrelated to its locus, ie context free. PERSPECT (Duffy and Kerr, 1993; Kerr, 1993) originally developed to demonstrate and evaluate the design utility of the customised viewpoints. Recently, Duffy and Duffy (Duffy and Duffy, 1996) utilised PERSPECT using the concept of controlled computational learning. PERSPECT is used to discuss the new

concept of shared learning and the coupling of designing and learning. PERSPECT is closely related in ethos with the situated learning in design, but significantly different. PERSPECT attempts to reflect the dynamic nature of learning in design through the generation of multiple and dynamic generalisations, to reflect and capture designers changing interests.

4. Situated learning about shape semantics

Shapes are fundamental to the act of designing in many fields. Through shapes designers express ideas and represent elements of design, abstract concepts and construct situations. Shapes denote edges and boundaries of spaces, building elements or abstract concepts. Hence, their role in designing is significant. In architectural design, as in many other design disciplines, shape composition is an important design activity. The formation and discovery of relationships among parts of a composition are fundamental tasks in designing (Mitchell and McCullough, 1991; Kolarevic, 1997). The relationships among shape parts can be of geometrical or non-geometrical. These relationships are called shape semantics. An architectural shape semantic is a collection of high-level information defining a set of characteristics with a semantic meaning based on a particular view of a shape such as reflective symmetry, rotation, repetition, or adjacency. Learning the interrelationships among these shape semantics is of importance in order to comprehend the architectural composition concepts. The act of designing is intrinsically dynamic in which design concepts and situations are constantly evolving. Hence, various relationships among them emerge in different representations during the design process. As designers draw and see what they have drawn, they make discoveries. They discover features and relations that cumulatively generate a fuller understanding of the configuration with which they are working. Perceptual interpretations are referred to as moves while the judgements of the consequences and implications of the move are referred as “seeing”. Different moves of the designers can yield an understanding of relationships, consequences and qualities of the design (Schon and Wiggins, 1992). The relationships between these shape semantics across different representations are the entities that constitute different design situations to situate these shape semantics in their environment. For a learning agent to be situated in the design process it should be to able learn the interrelationships among design knowledge and actions based on where they were used and be responsive to the changes that take place in the environment (Reffat and Gero, 1999).

5. Framework for situated learning in design

In this section, we describe the framework of a computational process model for **s**ituated **l**earning **i**n **de**sign called **SLiDe**. The role of SLiDe is to locate design knowledge, in the form of shape semantics, to its design situation by capturing the regularities of relevant relationships among the shape semantics across different representations, ie. learning the applicability conditions of design knowledge. SLiDe's framework consists of four modules: Generator, Recogniser, Situator and Situation Analyser as shown in Figure 1. The Generator module assists the designer to generate multiple representations of a single shape, in the form of floor plan. These representations can serve as a platform and form the environment for the Recogniser module to interact with and establish its own set of observations from the environment for the Situator module to learn from. The Recogniser module detects each representation and attributes shape semantics to it. The results of using the Recogniser module are a set of observations, each of which comprises a group of shape semantics associated with each corresponding representation within which they were used. The regularities of relationships among shape semantics at different observations, are the triggers for the Situator module to construct situational categories for these shape semantics. The situational categories are clustered based on the regularities of relationships among shape semantics based upon where they were used. The reason of calling them situational categories rather than normal categories or clusters as been used in machine learning is that knowledge is generalised with respect to the situation within which it was used. Updating what has been learned can be in the form of adding new knowledge that is relevant to a learned situation that was not previously available or constructing new situations. The results of SLiDe are sets of situational categories that situate the shape semantics. The following subsections present these four modules and their outcomes in more detail.

5.1 GENERATING MULTIPLE REPRESENTATIONS OF SHAPE

Multiple representations of design through re-interpretation are proposed as a platform for SLiDe. Multiple representations provide the opportunity for different shape semantics and relationships among them to be found from the image of a single object. This is important if these relationships are to be used later since it is not known in advance which of the possible relationships that could be formed are likely to be useful. Hence, multiple representations provide a platform for different situations to be encountered.

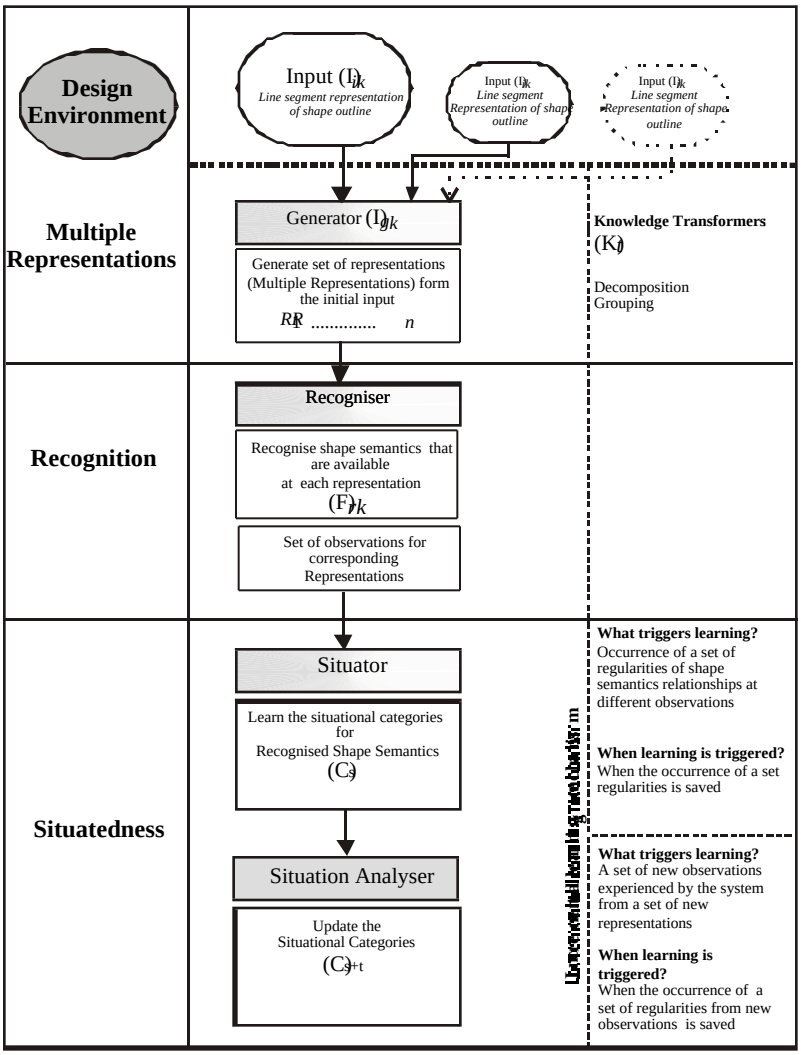

Figure 1. The overall architecture of the computational model of a system for situated learning in design

It appears that humans have no difficulty in using different representations for what is apparently the same object in order to achieve different goals. This fits well with the no-function-in–structure principle. The emergence of patterns in the re-interpretations of designs makes some shape semantics explicitly available for recognition. This contributes to making changes in what is available in a design environment by recognising shape semantics that were not explicitly

recognisable in the previous representations. An example of how a design object, square, can be seen and represented in many different ways is shown in Figure 2. A square can be represented as a set of four points; a set of four lines segments; a set of four infinite lines; and the perimeter of a given area or a region defined by four half planes. Different relationships might appear from these different representations. Some of the possible representations of an external representation of the square in Figure 2(a) are shown in Figure 2(b). Each of which is suitable for one or more application. The notion of multiple representations of a single object maps on the notion of a fluid world whereas most existing AI-based design systems view the world from a fixed viewpoint.

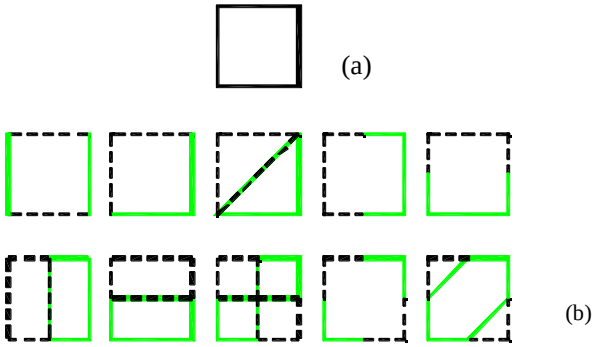

Figure 2 Some of the possible representations of a square

The Generator module handles the generation of different representations from what appears as a single object. The process model of the Generator module is shown in Figure 3. The Generator module commences with an initial representation of a shape in the form of line segments. It re-represents these line segments in the form of their infinite maximal lines. This module then requests the designer to select a shape of interest from among the intersections of infinite maximal lines. The Generator searches the design space for any shapes corresponding to the selected shape and generates a representation from the combination of corresponding shape and the boundary of the initial representation. If there are no corresponding shapes in the design space the designer is requested to have another selection. The Generator does not allow for overlapping shapes while searching for corresponding shapes. An example of generating multiple representations from a single shape representation using the Generator module is shown in Figure 4. The design description for the architectural plan of a library designed by Louis

Khan in New Hampshire (Clark and Pause, 1994) is shown in Figure 4(a). An initial representation of the architectural plan in the form of line segments is presented in Figure 4(b). The infinite maximal line method is used to re-represent the initial representation of the library, Figure 4(c). The results of using the Generator module to produce different representations from the initial representation are shown from Figures 4(d) to 4(l). This set of representations form the design environment for the Recogniser module to interact with and learn from. The Generator module is implemented using AutoCAD as the graphic interface.

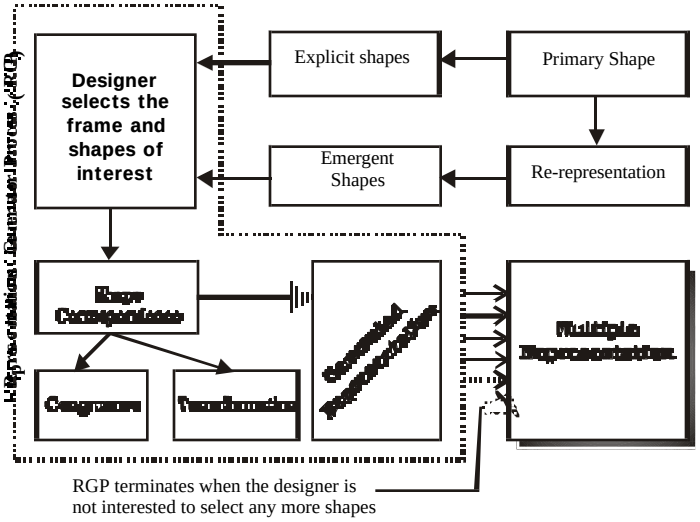

Figure 3. Process model of the Generator module

5.2 RECOGNITION OF SHAPE SEMANTICS

In this research, a drawing or part of it represents the design space. Within this design space various design qualities can be recognised or found. Since our interest here is in shape composition, the design qualities of interest within the design space are shape semantics. Shape semantics are visual patterns of relations among parts of the represented shape. Shape semantics have many characteristics; one of which is that they encapsulate design knowledge that appears in design artefacts and are among design knowledge that tend to be fundamental to aesthetic design. Shape semantics are recognised in terms of similarity of spatial relationships as well as physical properties. The Recogniser module uses a computable structural shape pattern representation that focuses on shape

relationships as well as physical properties (Cha and Gero, 1998). The Recogniser module provides the capability to recognise shape semantics from the multiple representations produced using the Generator module.

Shape semantics recognition starts from identification of shape congruency. Congruent shapes have the same structure of elements in terms of topology and geometry. If two shapes have the same number of infinite maximal lines, number of intersections, geometrical properties of infinite maximal lines and dimensional constraints of segments on each infinite maximal line, then these two shapes are congruent (Gero and Jun, 1995). This is to say that shapes are considered congruent if and only if structural properties of one shape are equivalent to structural properties of another shape in terms of topology and geometry. Shapes are represented as bounded polyline shapes. A bounded polyline shape is an enclosed polyline shape, for any point on the boundary of which there exists at least one circuit composed of line segments which starts from and ends at that point without covering any line segment more than once.

Shape semantics appear in architectural works. Some examples of shape semantics are reflective symmetry, repetition, adjacency, simple and cyclic rotations. Each shape semantic has preconditions without which it will not be recognised in a design space. For instance, the preconditions of reflective symmetry between two shapes are that they must be congruent and that certain geometric conditions are met by the midpoints of the lines joining corresponding vertices. The Recogniser module detects the shape semantics at each representation and produces an observation for each corresponding representation. Table 1 shows the set of observations (O_n) produced by the Recogniser module from the set of representations shown in Figure 4. In Table 1 $\boldsymbol{S_m}$, $\boldsymbol{P_r}$, $\boldsymbol{A_d}$, $\boldsymbol{R_c}$, $\boldsymbol{S_r}$ and $\boldsymbol{R_t}$ refer to reflective symmetry around multiple axes, repetition, adjacency, cyclic rotation, reflective symmetry and rotation respectively.

5.3 LOCATING SHAPE SEMANTICS WITHIN THEIR SITUATIONS

The Recogniser module finds each shape semantic when all of its preconditions are met but this says nothing about in which situation each shape semantic functions and operates. Each shape semantic has some relationships with other shape semantics that determine its applicability within the situation. The applicability conditions of shape semantics are the situated knowledge to be learned through the interrelationship between shape semantics within their environment. So, what is to be learned here is within which situation does each shape semantic operate. The Situator module locates the shape semantics within their situations by finding the regularities of relationships among them across the

observations. These regularities are arranged in terms of categories and since these categories reflect the relationships and dependencies among shape semantics based upon where they were used they are called situational categories.

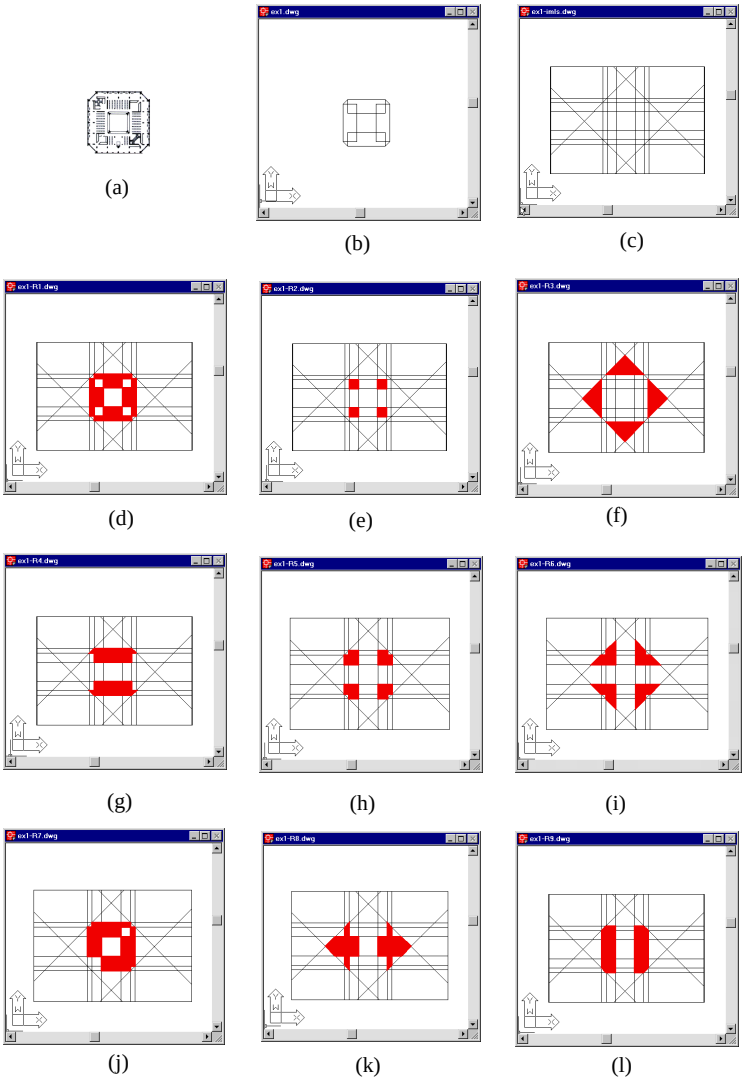

Figure 4. Some of possible representations developed using the Generator module: (a) design description of a library designed by Louis Khan, (b) an initial representation of the design description in the form of line segments, (c) infinite maximal line representation of the initial representation and from (d) to (l) a group of possible representations generated from the infinite maximal line representation.

Table 1. A set of observations produced using the Recogniser module to detect shape semantics available in the generated representations shown in Figure 4; $\boldsymbol{S_m}$, $\boldsymbol{P_r}$, $\boldsymbol{A_d}$, $\boldsymbol{R_c}$, $\boldsymbol{S_r}$ and $\boldsymbol{R_t}$ refer to reflective symmetry around multiple axes, repetition, adjacency, cyclic rotation, reflective symmetry and rotation respectively

Representation No.	Corresponding Observation (O_n)	
(d)	O_1	$\boldsymbol{S_m}$, $\boldsymbol{P_r}$, $\boldsymbol{A_d}$, $\boldsymbol{R_c}$
(e)	O_2	$\boldsymbol{S_m}$, $\boldsymbol{P_r}$, $\boldsymbol{A_d}$, $\boldsymbol{R_c}$
(f)	O_3	$\boldsymbol{S_m}$, $\boldsymbol{P_r}$, $\boldsymbol{A_d}$, $\boldsymbol{R_c}$
(g)	O_4	$\boldsymbol{S_r}$, $\boldsymbol{A_d}$
(h)	O_5	$\boldsymbol{S_m}$, $\boldsymbol{P_r}$, $\boldsymbol{A_d}$, $\boldsymbol{R_c}$
(i)	O_6	$\boldsymbol{S_m}$, $\boldsymbol{P_r}$, $\boldsymbol{A_d}$, $\boldsymbol{R_c}$
(j)	O_7	$\boldsymbol{A_d}$, $\boldsymbol{R_t}$
(k)	O_8	$\boldsymbol{S_r}$, $\boldsymbol{A_d}$
(l)	O_9	$\boldsymbol{S_r}$, $\boldsymbol{A_d}$

Within the situated view of designing, relationships and dependencies change over time whenever changes take place in the environment. Concept drift occurs when the environment changes. Concept drift implies that the clustering changes during the period of learning. The conditional probabilities reflected by the new observations change too and are no longer accurately represented by the categories in context-free machine learning systems. So the task of both the Situator and Situation Analyser modules can be defined as follows:

Initial input:

Given:
- a set of observations produced from multiple representations of an architectural plan, where each observation is comprised of a group of shape semantics recognised from a single representation

Find:
- clusterings that group these shape semantics across the observations into situational categories
- a summary description of each category that summarises its observations.
- a hierarchical organisation for those situational categories

Further Inputs:

Given:
- a new set of observations produced from other representations of the same architectural plan, where each

observation is comprised of a group of shape semantics recognised from a single representation

Find:
- reconstruct the learned clusterings to include the new observations within existing or new categories
- a summary description of each category that summarises its observations.
- update the hierarchical organisation of situational categories

To handle this task an incremental clustering mechanism that uses a nominal description of knowledge is needed. The role of this incremental clustering mechanism is to categorise observations, provide a description for each category, organise these categories hierarchically and most importantly not be affected by the order of observations as it deals with environments that change over time. This means that the learned categories are to be modified in response to the changes in the environment. In this task neither what is learned nor the environment that SLiDe interacts with are fixed. Both change over time as a reflection of the notion of situatedness.

The Situator and Situation Analyser modules in SLiDe model can be implemented using the modified unsupervised incremental clustering mechanism in COBBIT (Kilander and Jansson, 1993). The main reason for selecting COBBIT is that its focus on concept drift as well as its combination with COBWEB (Fisher, 1987) employ a unique clustering mechanism that fits the criteria of the task at hand. Overviews of clustering mechanisms as well as the unsupervised incremental clustering mechanism in COBWEB and COBBIT's modifications to it and the results of using COBBIT within the Situator and Situation Analyser modules are briefly presented in the following subsections.

5.3.1 An overview of clustering mechanisms

Clustering mechanisms automatically discover categories of observations that are similar along one or more dimensions. Once uncovered, these categories might suggest features that characterise observed knowledge. Ideally, clustering organises a space of observations in a way that facilitates more efficient problem solving. An observation might be a device specification that is matched against similar specifications; design decisions that were appropriate in previous cases that can be exploited as starting points for a new design.

It is not intended here to compare between different clustering mechanisms as that can be found elsewhere (Fisher and Schlimmer, 1988; Gennari et al, 1989; Fisher et al, 1993; Iba and Langley, 1999; Langley, 1999). The purpose of this overview is to investigate different clustering mechanism and select the mechanism that best meets the criteria of the

task at hand. Clustering has long been studied in numerical taxonomy, where observations are grouped and segregated based on numeric measures of similarity and dissimilarity. In many cases, though particularly in AI applications, symbolic and nominal data are abundant. Early work in conceptual clustering produced the Cluster system (Michalski and Stepp, 1983), which does not form classes based on similarity between observations, but seeks a partition whose categories are best described conjunctively, with a limited form of disjunction allowed. Each conjunction specifies attribute values that are true for all members of the corresponding cluster. In contrast to Cluster's conjunctive representation, COBWEB (Fisher, 1987) forms classes that may be best represented probabilistically, described by probability distributions of the attribute values exhibited by their members. Like Cluster, COBWEB associates interpretations with clusters, but their probabilistic representations are more relaxed than those of Cluster's conjunctive scheme. COBWEB is an incremental unsupervised clustering mechanism. Like COBWEB, the Autoclass system (Cheeseman and Stutz, 1995) is a clustering method that represents clusters probabilistically. It differs from COBWEB in that it takes a Bayesian position in the classification and incorporates a probabilistic variant of the non-incremental algorithm known as expectation maximisation (Iba and Langley, 1999). COBBIT (Kilander and Jansson, 1993) is a variant on COBWEB designed explicitly to deal with environments that change over time and its clustering structures are not dependent on the order of the observations.

5.3.2 Unsupervised incremental clustering mechanism in COBWEB

Unsupervised learning means that observations are clustered in categories without advice from a teacher. In other words, the learning mechanism includes not only deciding which observations each category should contain, but also the number of such categories. The learning implemented in COBWEB uses an incremental hill-climbing learner. Incremental hill-climbing searches an n-dimensional space over which some function f is defined. This function determines the shape of the n-dimensional surface and the agent attempts to find that point with the highest f score. New observations modify the contours of the surface. The hills and valleys of the incremental hill-climbing learner's space are constantly changing as it gets more new knowledge.

COBWEB uses a slightly generalised version of Gluck and Corter's category utility (Gluck and Corter, 1985) as an evaluation function to control its classification and learning behaviour. Category utility favours clustering that maximise the potential of inferring information. In doing this, it attempts to maximise intra-class similarity, predictability, and inter-class differences, predictiveness, and it also provides a principled

$$\frac{\sum_{k=1}^{n} P(C_k) \sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2}{n} \quad (1)$$

tradeoff between both of them. For any set of observations, any attribute-value pair, $A_i = V_{ij}$, and any class C_k, one can compute P ($A_I = V_{ij}|C_k$), the conditional probability of the value given membership in the class, or its predictability. One also can compute P ($C_k|A_i = V_{ij}$), the conditional probability of membership in the class given this value, or its predictiveness. The category utility is defined as the increase in the expected number of attribute values that can be correctly guessed, given a set of n categories, over the expected number of correct guesses without such knowledge. The complete expression for category utility is as shown in expression (1) (Fisher, 1987).

The ability to achieve high quality descriptions for categories, despite the limitations of hill-climbing such as the tendency to halt at local optima and dependence on step size, is maintained by extending the set of available operators. Rather than restricting search to be unidirectional, both generalisation and specialisation operators are supplied. Bidirectional mobility allows such an incremental system to recover from a bad learning path. COBWEB can invoke four operators to alter the structure of its clustering's hierarchy. As illustrated in Figure 5, these include (Iba and Langely, 1999):

a. extending downward, which occurs when a training set (set of observations) reaches a terminal node in memory
b. creating a disjunct, creates a new child based on this recent set, which occurs if the training set is sufficiently different from all the children of a node
c. merging two categories, which occurs if a set is similar enough to two children of a node
d. splitting a category, which occurs when a child of a node no longer serves as a useful category.

COBWEB considers the last three of these actions at each level of the hierarchy, as it sorts the new training observation downward through memory.

5.3.3 COBBIT's modification to COBWEB

COBWEB is designed to work under a condition of clustering constancy that does not distinguish between new and old observations, requiring new features to replace previous ones by quantity, just as most other machine learning systems. When COBWEB is incrementally and sequentially exposed to the extensions of a set of clusters, it retains all observations,

disregards the age of a category and may create different categorical structures dependent on the order of the observations. These three characteristics make COBWEB sensitive to the effects of concept drift. COBBIT (Kilander and Jansson, 1993) is a variant on COBWEB designed explicitly to overcome these limitations. It deals with environments that change over time and its clustering structures are not dependent on the order of the observations.

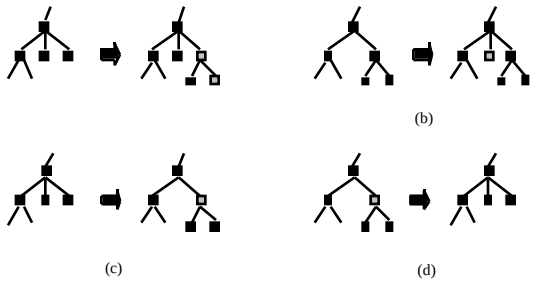

Figure 5. Learning operators used to modify the structure of a hierarchy of probabilistic clustering: (a) extending the hierarchy downward; (b) creating disjunct at an existing level; (c) merging two existing classes; and (d) splitting an existing category. Newly created nodes are shown in grey (Iba and Langely, 1999)

After a basic hierarchy is established in COBWEB, further input either confirms what is already learned or one of the following two cases: a new observation that adds knowledge, because there is still more to learn or a new observation that replaces old knowledge, because the domain is evolving. COBBIT's modification to COBWEB resides in the control system, the way learning, predictive performance and training are combined to form a complete system. COBBIT equipped COBWEB with the dynamic deletion of old observations using a queue of observations and continuous monitoring of performance. The COBWEB standard control loop (read-learn) has been extended in COBBIT to be (read-evaluate-learn-trim) in which control parameters of the upper and lower bounds on the number of elements in the queue at any time, update time, are set by the user. A continuous monitoring of performance algorithm is implemented by having a queue of training observations and dynamically altering the size of the queue depending on its performance. As each observation is presented to COBBIT, it attempts to predict each and every attribute that has a known value. The percentage of correctly predicted attributes is an output of the current performance index. The trend of this index allows for corrective action as soon as a drop in the performance is observed. The performance index is used to determine the size of the queue. This behaviour is intended to remove old observations with low performance index from the hierarchy. Also, by using the queue

mechanism and the subtraction of low performance index observations, ordering effects in the category hierarchies are only applicable to the training observations in the queue since COBBIT does not alter the input ordering in any way. In COBBIT, categories are continuously influenced by recent observations to confirm (reinforce) or degrade (decay) learned categories or create new ones.

5.3.4 Results of using the Situator module

The Situator module employs COBBIT as its unsupervised clustering mechanism to search for the regularities in the observations and categorise them in the form of situational categories $\mathbf{C}_s$. The results of using the Situator module are shown in Figure 6. The graph shown in Figure 6 is one of the automated forms produced by the Situator module representing the hierarchical structure of situational categories. Each situational category is associated with a summary description that summarises its observations.

In Figure 6, the situational category $\mathbf{C}_{s2}$ represents the regularity across the observations of relationships among shape semantics $\boldsymbol{S_m}$, $\boldsymbol{P_r}$, $\boldsymbol{A_d}$ and $\boldsymbol{R_c}$. Within this regularity if the current knowledge goal is $\boldsymbol{S_m}$, reflective symmetry, then the other parts of the regularity $\boldsymbol{P_r}$, $\boldsymbol{A_d}$ and $\boldsymbol{R_c}$ construct the situation within which $\boldsymbol{S_m}$ operates and functions. So, $\boldsymbol{S_m}$ is situated within these shape semantics. On the other hand, if $\boldsymbol{R_c}$ is the knowledge in focus, then the other parts of the regularity $\boldsymbol{S_m}$, $\boldsymbol{A_d}$ and $\boldsymbol{P_r}$ construct the situation within which $\boldsymbol{R_c}$ operates and functions. This is a duality between parts of the regularity, ie duality between knowledge and situation as shown in Figure 7.

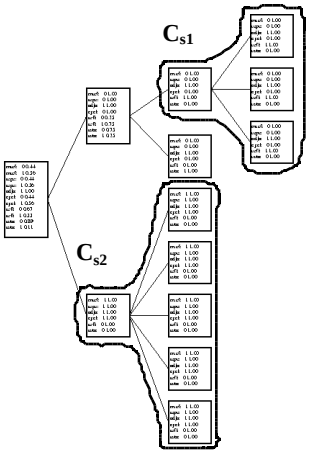

Figure 6. Two situational categories learned by the Situator module from the set observations produced by the Recogniser module and shown in Table 1

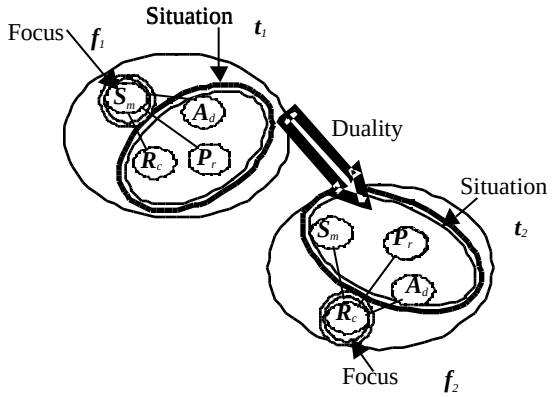

Figure 7. The duality between knowledge and situation within the situational category $\mathbf{C_{s2}}$

5.3.5 Results of using the Situation Analyser module

The Situation Analyser is triggered to update the learned situational categories whenever the Recogniser module detects any new observations from the design environment. The Situation Analyser facilitates the incremental clustering mechanism in COBBIT to update what has been learned. For instance, if a new set of two observations is experienced in SLiDe, The Situation Analyser will analyse the learned knowledge taking into account the new observations and try either to fit them within the existing categories or create new categories or sub categories to accommodate them. Figure 8 shows the learning results produced by the Situation Analyser module after a new set of observations. It created a new category $\boldsymbol{C_{s3}}$ that categorised both of the two new observations in one new situational category based upon the regularities of relationships among shape semantics available on them and the their distinctive features from the situational category that were learned previously. When the Situation Analyser is exposed to another new set of observations consisting of four observations, it restructured its learned knowledge as can be seen in Figure 9 where two new subsets of the previously learned situational categories $\boldsymbol{C_{s2(a)}}$ and $\boldsymbol{C_{s3(a)}}$ emerge. The restructuring of situational categories in Figure 8 can be seen as a mere adding of a new situational category to the existing ones. On the other hand, as can seen in Figure 9 there is an overall restructuring that has included the additional observations and accommodated them in the form of subcategories within the previously learned situational categories.

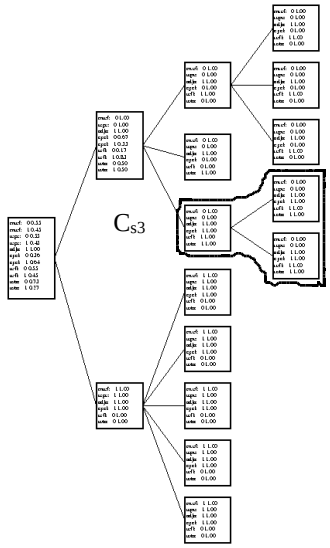

Figure 8. *A newly created situational category ($\mathbf{C_{s3}}$) by the Situation Analyser in response to an additional set of observations*

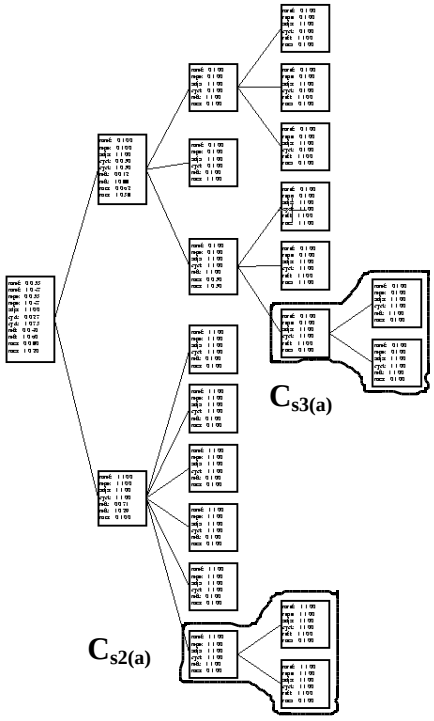

Figure 9. An overall restructuring whereby two sub-situational categories ($\mathbf{C_{s2(a)}}$ and $\mathbf{C_{s3(a)}}$) are emerged in response to the most recent additional set of observations

5. Discussion:

Current CAD systems can only be used at the latter stages of the design process after most of the major design decisions have been made. At the same time they are passive systems in the sense that they do not modify their behaviours in response to the changes in the design environment. The ability to bring CAD systems into the early stages of designing and to provide useful designing support at the conceptual stages is important in design computing. Such support creates the opportunity for CAD systems to provide designers with useful and applicable design knowledge during the generation of design concepts. The usefulness and applicability of such knowledge is based on its operational significance and applicability to the situation at hand.

We have developed a computational model for situated learning in design (SLiDe). SLiDe is a system that locates design knowledge in relation to its situation based upon where it was used and applied. It modifies its behaviour as its situation from the design environment changes. SLiDe is an active system that responds to dynamic changes in its environment. It selects appropriate actions as a response to its immediate situation through recognising various contexts to which it is potentially situated. SLiDe structures its encountered situations and classifies them into situational categories in a hierarchical manner.

Developing a computational model of situated learning in design to produce these kinds of situational categories provides opportunities to assist designers during the conceptual design process. One way is to integrate SLiDe with current CAD systems such as AutoCAD to make it easier for designers to conceptualise and explore their designs beyond the mere drafting of them as AutoCAD is currently used. SLiDe helps to explore shapes of designed elements drawn in AutoCAD and allows the designer to have varieties of representations of what he/she has designed that may lead the designer to a different move. These different representations of the same design help to arouse the designer's attention to potentially hidden visual features of his/her design elements. This can be called, enhancing the perceptual interaction with design elements. Further, SLiDe can excite designer's attention to a set of shape semantics available in his/her current design by highlighting a particular set of design elements that reflect those semantics that the designer might have indicated attracted him/her. Designers tend to lose track of what attracted them earlier as they proceed and make changes in their designs. SLiDe, having stored the designer's interest as the focus can dynamically change the association between design elements during the design process by maintaining the situation of the designer's focus. So, whenever the designer made changes in the design after indicating the semantics of

interest, SLiDe can change all the interrelated design elements to maintain the focus by maintaining the relationships that define the situation of that focus. Using SLiDe to provide such a features in current CAD systems can potentially help to support designers in designing as well as drafting and at the same time will change the nature of current CAD systems from passive systems to active support design systems. Such systems learn in a situated approach and apply what has been learned based on the situations they encounter.

Acknowledgments

This research is supported by a University of Sydney Postgraduate Award and by an Australian Research Council grant. The authors acknowledge the comments of the three anonymous referees of this paper

References

Akman, V. and Surav, M.: 1995, Contexts, oracles, and relevance, *AAAI-95 Fall Symp. on Formalizing Context*, Cambridge, Mass.

Balsam, P. D.: 1985, The function of context in learning and performance, *in* P. D. Balsam and A. Tomie (eds), *Context and Learning*, L. Erlbaum Associates, Hillsdale, NJ, pp. 1-21.

Barwise, J. and Perry, J.: 1983, *Situations and Attitudes*, The MIT Press, Cambridge, Massachusetts.

Cha, M. and Gero, J. S.: 1998, Shape pattern recognition using a computable pattern representation, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '98*, Kluwer Academic Publishers, The Netherlands, pp. 169-187.

Cheeseman, P. and Stutz, J: 1995, Bayesian Classification (AutoClass): Theory and Results, *in* U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy (eds), *Advances in Knowledge Discovery and Data Mining*, The AAAI Press, Menlo Park.

Clancey, W. J.: 1995, A tutorial on situated learning, in J. Self (eds), *Proceedings of the International Conference on Computers and Education (Taiwan)*, AACE, Charlottesville, VA, pp. 49-70.

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

Clark, R. and Pause, M.: 1996, *Precedents in Architecture*, Van Nostrand Reinhold, New York.

Duffy, A. H. B. and Kerr, S. M.: 1993, Customised perspectives of past design from automated group rationalisations, *Artificial Intelligence in Engineering*, **8**(3): 183-200.

Duffy, S. M. and Duffy, A. H. B.: 1996, Sharing the Learning Activity using Intelligent CAD, *Artificial Intelligence for Engineering Design, Analysis and Manufacture (AI EDAM)*, **10**(2): 83-100.

Fisher, D. and Schlimmer, J.: 1988, Models of incremental concept learning, *Technical Report* 88-05, Department of Computer Science, Vanderbilt University, Nashville, TN.

Fisher, D., Xu, L., Carnes, J., Reich, Y., Fenves, S., Chen, J., Shiavi, R., Biswas, G. and Weinberg, J.: 1993, Analysing AI clustering to engineering tasks, *IEEE Expert*, **8**, 51-60.

Fisher, D.: 1987, Knowledge acquisition via incremental conceptual clustering, *Machine Learning*, **2**, 139-172.

Gedenryd, H.: 1998, How Designers Work, *Ph.D Thesis*, Lund University, Lund.

Gennari, J., Langley, P. and Fisher, D.: 1989, Models of incremental concept formation, *Artificial Intelligence*, **40**, 11-62.

Gero, J. S. and Jun, H. J.:1995, Getting computers to read the architectural shape semantics of drawings, *ACADIA '95*, pp. 97-112.

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

Gero, J. S.: 1999, A model of designing that includes its situatedness, in J. Gu and Z. Wei (eds), *CAADRIA'99*, Shanghai Scientific and Technological Literature Publishing House, Shanghai, China, pp. 235-242.

Gluck, M. and Corter, J.: 1985, Information, uncertainty and the utility categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, Irvine, CA, pp. 283-287.

Iba, W. and Langley, P.: 1999, Unsupervised learning of probabilistic concept hierarchies. *Unpublished manuscript*, Institute for the Study of Learning and Expertise, Palo Alto, CA.

Kerr, S.: 1993, Customised viewpoint support for the utilisation of experiential knowledge in design, *Ph.D. Thesis*, CAD Centre, Department of Design, Manufacturer and Engineering Management, University of Strathclyde, Glasgow, UK.

Kilander, F. and Jansson, C.: 1993, COBBIT: a control procedure for COBWEB in the presence of concept drift. *Proceedings of the 1993 European Conference on Machine Learning*, Springer-Verlag, Vienna, pp. 244-261.

Kolarevic, B.: 1997, Regulating lines and geometric relations as a framework for exploring shape, dimension and geometric organisation in design, *in* R. Junge (ed.), *CAAD Futures 1997*, pp. 163-170.

Langley, P.: 1999, Concrete and abstract models of category learning. *Proceedings of the Twenty-First Annual Conference of the Cognitive Science Society*. Vancouver, BC: Lawrence Erlbaum.

Matwin, S. and Kubat, M.: 1996, The role of context in concept learning, *ICML-96 Workshop on Learning in Context-Sensitive Domains, at the 13th International Conference on Machine Learning*, Bari, Italy.

Merriam-Webster's Online, http://www.m-w.com/.

Michalski, R. and Stepp, R.: 1983. Automated construction of classification: conceptual clustering versus numerical taxonomy, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **5**(4), 396-409.

Mitchell, W. and McCullough, M.: 1991, *Digital Design Media*, Van Nostrand Reinhold, New York.

Nardi, B. A.: 1996, Studying context: A comparison of activity, theory, situated action models and distributed cognition, *in* B. A. Nardi (ed.), *Context and Consciousness: Activity Theory and Human-Computer Interaction*, MIT Press, Cambridge, Mass, pp. 69-102.

Radvansky, G. and Zacks, R.: 1997, The retrieval of situation-specific information, in M. Conway (ed.), *Cognitive Models of Memory*, The MIT Press, Cambridge, Massachusetts, pp. 173-213.

Reffat, R. M. and Gero, J. S.: 1999, Situatedness: A new dimension for learning systems in design, *in* A. Brown, M. Knight and P., Berridge (eds), *Architectural Computing: from Turing to 2000*, eCAADe and The University of Liverpool, UK, pp. 252-261.

Reich, Y. and Fenves, S. J.: 1992, Inductive learning of synthesis knowledge, *International Journal of Expert Systems: Research and Applications*, **5**(4): 275-297.

Reich, Y.: 1991, Constructive induction by incremental concept formation, in Y. A. Fledman and A. Bruckstein (eds), *Artificial Intelligence and Computer Vision*, Elsevier Science Publishers, Amsterdam, pp. 191-204.

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

Schlimmer, J. C. and Granger, R. H.: 1986, Beyond incremental processing: Tracking concept drift, *Proceedings of AAAI-86*, Mrgan Kaufmann, Philadelphia, PA, pp. 502-507.

Schon, D. and Wiggins, G.: 1992, Kinds of seeing and their functions in designing, *Design Studies*, **13**(2),135-156.

Widmer, G. and Kubat, M.: 1996, Learning in the presence of concept drift and hidden contexts, *Machine Learning*, **23**(1): 69-101.

This paper is a copy of: Reffat, R. and Gero, J. S. (2000) Computational situated learning in design, *in* Gero, J. S. (ed.), *Artificial Intelligence in Design'00,* Kluwer, Dordrecht , pp. 589-610.