Computational Situated Learning in Designing

Application to Architectural Shape Semantics

by

Rabee M. Reffat

B.Arch (Hons), M.Sc. Arch. Eng.

A Thesis Submitted for the Degree of
Doctor of Philosophy

Department of Architectural and Design Science
Faculty of Architecture
University of Sydney

Dedication

This thesis is dedicated to the loving memory of my beloved father.
My Lord bestow your mercy on him as he cherished me in my childhood.

Acknowledgments

I would like to express my sincere appreciation and gratitude to the people in the Key Centre of Design Computing and Cognition, KCDCC, for their encouragement and support. First, I would like to thank my supervisor Professor John S. Gero for his continuous support in my Ph.D. program. Professor Gero introduced me to a hybrid world of design computing, artificial intelligence and cognitive science. He taught me how to express ideas, approach a research problem and the need to be persistent to accomplish my goals. His patient support during the last four years helped me to bring the research presented in this thesis to a successful conclusion.

A special thanks goes to Professor Mary Lou Maher and Dr. Mike Rosenman for their valuable comments on the thesis proposal and to Professors Nigel Cross and Tim Smithers for discussing my research with me during their visit to KCDCC. I would like to thank the faculty members and staff at KCDCC especially Dr. Scott Chase, Dr. Simeon Simoff, Paul Murty, Fay Sudweeks, Dr.Vladmir Kazakov, Dr. Masaki Suwa, Dr. Manolya Kavakli, Joe Nappa, Doug Scoular, Andrew Winter, Anne Christian and my colleagues Dr. Jose Damski, Dr. Josiah Poon, Dr. Thorsten Schnier, Dr. Anna Cicognani, Dr. Lan Ding, Dr. Soohoon Park, Dr. Andres Gomez, Philip Tomlinson, Katy Bridge, Gourabmoy Nath, Gerard Gabriel, Guang Shi, Jaroslaw Kulinski, Robert Saunders, Fei Li, Hsien-Hui Tang, Chin Chin Kau, Eonyong Kim, Ellina Yukhina, Justin Clayden, Greg Smith and Stephen Clarke. I am grateful for the financial support of both the Key Centre of Design Computing and Cognition and the University of Sydney scholarships. My enormous appreciation belongs to Dr. Edward L. Harkness for proofreading and reviewing the English in this thesis.

The heart of my dedication belongs to my beloved family: my mother, brothers and sisters, my wife and my son for the depth of their love. I am indebted to my wife who devoted her life to our small family lovingly and willingly. The presence of my family and my circle of friends, new and old, near and far, provided me with passion, support and identity.

Summary

Learning the situatedness (applicability conditions), of design knowledge recognised from design compositions is the central tenet of the research presented in this thesis. This thesis develops and implements a computational system of situated learning and investigates its utility in designing. Situated learning is based on the concept that "knowledge is contextually situated and is fundamentally influenced by its situation". In this sense learning is tuned to the situations within which "what you do when you do matters". Designing cannot be predicted and the results of designing are not based on actions independent of what is being designed or independent of when, where and how it was designed. Designers' actions are situation dependent (situated), such that designers work actively with the design environment within the specific conditions of the situation where neither the goal state nor the solution space is completely predetermined. In designing, design solutions are fluid and emergent entities generated by dynamic and situated activities instead of fixed design plans. Since it is not possible in advance to know what knowledge to use in relation to any situation we need to learn knowledge in relation to its situation, ie learn the applicability conditions of knowledge. This leads towards the notion of the situation as having the potential role of guiding the use of knowledge.

Situated **L**earning **i**n **De**signing (**SLiDe**) is developed and implemented within the domain of architectural shape composition (in the form of floor plans), to construct the situatedness of shape semantics. An architectural shape semantic is a set of characteristics with a semantic meaning based on a particular view of a shape such as reflection symmetry, adjacency, rotation and linearity. Each shape semantic has preconditions without which it cannot be recognised. Such preconditions indicate nothing about the situation within which this shape semantic was recognised. The situatedness or the applicability conditions of a shape semantic is viewed as, the interdependent relationships between this shape semantic as the design knowledge in focus, and other shape semantics across the observations of a design composition. While designing, various shape semantics and relationships among them emerge in different representations of a design composition. Multiple representations of a design composition by re-interpretation have been proposed to serve as a platform for SLiDe. Multiple representations provide the opportunity for different shape semantics and relationships among them to be found from a single design composition. This is important if these relationships are to be used later because it is not known in advance which of the possible relationships could be constructed are likely to be useful. Hence, multiple representations provide a platform for different situations to be encountered. A symbolic representation of shape and shape semantics is used in which the *infinite maximal lines* form the representative primitives of the shape.

SLiDe is concerned with learning the applicability conditions (situatedness), of shape semantics locating them in relation to situations within which they were recognised (situation dependent), and updating the situatedness of shape semantics in response to

new observations of the design composition. SLiDe consists of three primary modules: Generator, Recogniser and Incremental Situator. The Generator is used by the designer to develop a set of multiple representations of a design composition. This set of representations forms the initial design environment of SLiDe. The Recogniser detects shape semantics in each representation and produces a set of observations, each of which is comprised of a group of shape semantics recognised at each corresponding representation. The Incremental Situator module consists of two sub-modules, Situator and Restructuring Situator, and utilises an unsupervised incremental clustering mechanism not affected by concept drift. The Situator module locates recognised shape semantics in relation to their situations by finding regularities of relationships among them across observations of a design composition and clustering them into situational categories organised in a hierarchical tree structure. Such relationships change over time due to the changes taken place in the design environment whenever further representations are developed using the Generator module and new observations are constructed by the Recogniser module. The Restructuring Situator module updates previously learned situational categories and restructures the hierarchical tree accordingly in response to new observations.

Learning the situatedness shape semantics may play a crucial role in designing if designers pursue further some of these shape semantics. This thesis illustrates an approach in which SLiDe can be utilised in designing to explore the shapes in a design composition in various ways; bring designers' attention to potentially hidden features and shape semantics of their designs; and maintain the integrity of the design composition by using the situatedness of shape semantics. The thesis concludes by outlining future directions for this research to learn and update the situatedness of design knowledge within the context of use; considering the role of functional knowledge while learning the situatedness of design knowledge; and developing an autonomous situated agent-based designing system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents a brief prologue; introduces the motivation; outlines the aims and objectives of the research presented in this thesis; presents an overview of the scope; and concludes with a brief outline of the chapters that follow.

In this thesis, the word "designing" is used to refer to the activity of making designs and "design" refers to the product of designing. In designing, the problem is ill-defined whereby design actions are not based on performing a complete plan or a program that is given a priori or at the beginning of designing. All relevant information cannot be predicted and established in advance of the design activity. The directions that are taken during the exploration of the design territory are influenced by what is learned along the way and the partial glimpses of what might lie ahead (Cross, 1999). In designing, both the problem and solution spaces co-evolve, constantly being revised. The focus of attention shifts back and forth from defining the problem to solving the problem. Designing involves making decisions and even making decisions about decisions, ie what actions to execute first. Furthermore, the result of designing is not based on actions independent of what is being designed or independent of when, where and how it is being designed.

The empirical approach of design studies looks carefully at a specific design process aimed at understanding design activities in terms of actions and decisions taken. In these decisions the process and content of the design process are often inextricably linked. Schön (1983) based on his experiment with a particular example of architectural design explored designing as a "conversation with the materials of the situation" where the design process is viewed as reflection-in-action where designers deal with situations. Suwa et al (1998b) from a macroscopic analysis of design processes based on a scheme for coding designers' cognitive actions concluded that “sketches serve as a physical setting in which design thoughts are constructed on the fly in a situated way”. These results coincide with the views of Agre and Chapman (1987) and Kirsh (1995) in which people act not just in goal-oriented or knowledge-intensive ways, but more often in response to visuo-spatial features of the physical setting they are in. From another cognitive perspective, Gedenryd (1998) perceived designers' actions as situation dependent (situated), such that designers work interactively with the design environment within the specific conditions of the situation. Gero (1998a) viewed conceptual designing as a sequence of situated acts. Most recently, tools from cognitive science have started

to provide insights into human designing (Lawson, 1980; Akin, 1986; Cross et al., 1996; Gedenryd, 1998; Morrison, 1998; Lueg, 1999).

Artificial intelligence in design is aimed at supporting designing through developing systems or tools that either aid designers or emulate designing, that is, developing autonomous designing tools. Knowledge of the human designer's cognitive behaviour obviously is of fundamental importance, because the users of such tools (designers), must be able to use them in ways that are cognitively comfortable. So the systems must be designed on the basis of models of the cognitive behaviour of the systems' users (Cross, 1999).

1.1 Motivation

There are three primary contributions that motivated the research presented in this thesis: situatedness of designing, situatedness of learning and the role of situatedness in learning in designing.

- **Situatedness of designing**

Design cannot be predicted and the designer has to be 'at a particular set of states' in order to decide what to do. The acquisition of design knowledge is not a stored body of data, but rather a capability constructed in action within the situation. This leads us towards the notion of situation as having the potential role of guiding the use of knowledge and designing as a situated and dynamic activity. Since it is not possible to know beforehand what knowledge to use in relation to any situation we need to learn knowledge in relation to its situation. 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 (Clancey, 1997b) acknowledges 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 in designing had been based on the view of knowledge being unrelated to either its locus or application (Gero, 1999). Similarly, the vast majority of machine learning approaches and applications in designing have made the implicit assumption that "what has been learned is potentially universally applicable". This assumption can be restated as "the context within what is learned plays no role" (Gero, 1998b). On the contrary, situatedness in designing is concerned with locating design knowledge in relation to its situation within the design environment. One of the important characteristics of situatedness in designing is the dynamic change in the design context while designing. The whole design context constitutes the design environment and for a specific knowledge in focus the active and immediate part of that context (the situation), plays a significant role.

The situatedness of an object acts as an operator that locates knowledge in relation to its situation. For instance, consider the difference between a table in an office space and one in a dining room. A table placed in an office space has different relationships to its surroundings than the one placed in a dining room. In an office space there are relationships between the table and filing cabinets, drawers, desk lamp, computer, chairs and their arrangement that structure this table in relation to its surroundings to be

recognised as a desk or an office table. On the other hand, in a dining room the relationships between the table (although it might be the same object), certain objects in its surroundings and the arrangement of chairs around it make it considered to be a dining table. So, for a system to locate such an object in relation to its situation it needs to learn the regularities of its relationships to its surroundings within which it was recognised.

Design context, ie situations, are embedded in design drawings (Do, 1998). Drawings provide architects with a medium to express their design concepts (Robbins and Cullinan, 1994). In architectural drawings, shapes are fundamental to the act of designing. Designers express ideas and represent elements of design using shapes, 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, 1995; Kolarevic, 1997). The relationships among shape parts can be geometrical or non-geometrical in nature. These relationships are called shape semantics. An architectural shape semantic is a set of characteristics with a semantic meaning based on a particular view of a shape such as reflective symmetry, rotation, repetition and adjacency. The relationships among shape semantics are reflections of designers' situated actions that led to different shape compositions. For a learning system to locate these shape semantics in their situations it should be capable of learning the regularities of relationships among these shape semantics from its own observations of the design composition.

The act of designing is intrinsically dynamic in which design concepts and situations are constantly evolving. Hence, various relationships among shape semantics emerge in different representations during the process of designing. 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. Different moves of the designers can yield an understanding of relationships, consequences and qualities of the design (Schön and Wiggins, 1992). Multiple representations through re-interpretations of designs could help in making some shape semantics that were implicit to be explicit. This contributes to constructing new observations of the design composition by recognising the shape semantics that were not explicitly recognisable in the previous representations. The notion of multiple representations from a single shape fits well here to provide a system with the opportunity to recognise different shape semantics and relationships among them from the image of an external representation, eg a drawing of an architectural plan. 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 relationships to be encountered. The regularities of relationships among these recognised shape semantics across different observations are the entities that constitute different design situations.

Intuitively, the notion of "environment" in AI refers to the relatively enduring and stable set of circumstances that surround an agent. The environment is where an agent lives and

influences its actions (Agre and Horswill, 1997). Since the focus of this thesis is system-based rather than agent-based, the definition of the "environment" refers to the internal model constructed by the system within which to learn. The initial set of representations developed by the system constitutes its design environment and changes take place in the environment whenever further representations are generated. So, it is a kind of internal environment within which the system recognises shape semantics. The system constructs its own set of observations and locates shape semantics in relation to their situations encountered from this environment. Changes in the environment give rise to the learning system to change its own observations and its behaviour accordingly. The system dynamically updates the situatedness of its learned knowledge either by refining the learned situation or constructing new situations to accommodate the changes in the environment.

- **Situatedness of learning in designing**

All learning occurs in context, ie situation (Balsam, 1985). The term context means the general conditions or circumstances in which an event takes place (Akman and Surav, 1996). Learning occurs in a context that is defined by specific features of the task at hand. These features make the learning context-sensitive. Many learning applications necessitate the use of context in learning (Matwin and Kubat, 1996). Learning systems that can both identify and manage the context will have a substantial advantage over a system that can manage context but requires a human operator to identify context (Turney, 1996). Learning and designing are closely related activities. Designers learn when they encounter knowledge that is sufficiently different from their present state of knowledge (Persidis and Duffy, 1991; Duffy and Duffy, 1996b). Learning in designing can be viewed as acquiring knowledge associated with its situatedness and learning is incremental and evolutionary, involving internal adjustment. Situation is conceived as the immediate context for the acquisition of that knowledge from the environment. What makes one situation different from or similar to others are the relationships that allow relevant distinctions to be made among situations.

Situated learning in designing takes as its focus the relationships between knowledge and the situation within which it occurs. Situated learning joins a growing literature in cognitive studies, discourse analysis and sociolinguistics where the common element is the premise that learning is defined relative to context, not self-contained structures (Lave and Wenger, 1991). The conception of situated learning is more encompassing in intent than conventional notions of learning in situ or learning by doing. Lave and Wenger (1991) explored learning as "legitimate peripheral participation" where peripherality is a positive term, whose most salient conceptual antonyms are un-relatedness or irrelevance to ongoing activity. The implication of situated learning in designing is that what is learned and experienced depends on the situation and the situation is not simply a variable to be manipulated, but it is constructed in an interactive ongoing manner.

- **The role of situatedness in learning in designing**

The concept of situatedness within learning and designing could be applied in developing a computational system for situated learning in designing by learning the relationships

between design knowledge and its situation and responding to the changes that take place in its environment.

In designing, knowledge is composed such that subsequent experiences categorise what was experienced before. How it is categorised depends on the ongoing sequence in which it becomes a part and its appropriateness is determined by constraining the domain in which categorisation occurs. The categorisation of the relationships among design knowledge (shape semantics), is based on the regularities in the design environment. Such regularities of relationships form the ground to situate that knowledge. The view of situated learning in designing is concerned with finding the regularities of relationships among shape semantics in relation to the situation within which they were recognised and construct the categories or clusters that accommodate these regularities. The learned categories carry with them notions of the applicability conditions of knowledge derived from the situation. These applicability conditions (situatedness) are then modified over time as changes take place in the design environment.

The approach of situated learning in designing has the prospect for making the computational-learning situation-dependent and not context free or universally applicable. The dependency on the situation (situatedness), implies the acquisition of the design knowledge within which it was recognised. This denotes the possibility of changing what has been learned depending on the encountered situations. The change is in the sense that the learning outcome, after modifying or changing the environment, is not constant. This would provide designers with systems that reflect the concept of designing as not being predetermined but dynamic. This would broaden the view for designers about the design environment and potentially guide the use of knowledge.

1.2 Aims and Objectives

The main goal of this thesis is to develop an approach to situated learning in designing in which computers will have the capability to learn design knowledge including concepts of its situatedness. This computational approach for situated learning in designing is implemented in a system called **SLiDe (S**ituated **L**earning **i**n **De**signing). SLiDe is a system that locates knowledge to its locus and application and modifies the situatedness of knowledge as its design environment changes. SLiDe has the capability to recognise various contexts in which it is potentially situated. SLiDe structures its encountered situations and classifies them into categories in a hierarchical structure tree. These categories carry with them the applicability conditions of design knowledge derived from the situation. SLiDe is implemented and exemplified within the domain of architectural shape semantics. SLiDe could be integrated into conventional CAD systems within the domain of designing architectural shape composition (SLiDe-CAAD), to provide interactive designing support at the conceptual stages. Four major objectives are to be achieved through the development of this thesis:

(a) *To develop multiple representations from an external representation of a design composition (floor plan), to constitute the design environment within which SLiDe is to learn the situatedness of design knowledge. Put differently, to develop a platform for a situated learning system in designing.*

This objective can be achieved by using infinite maximal lines to represent the shape of a design composition. Commencing with an initial representation of a shape in the form of line segments, these line segments are re-represented in the form of their infinite maximal lines (Gero, 1992b; Gero and Yan, 1993). The intersections of the infinite maximal lines define the design space to be searched. Then the designer selects a shape of interest from among the intersections of infinite maximal lines and the design space is searched for shapes that correspond to the selected shape (without allowing for overlapping shapes while searching). If corresponding shapes are found, a new representation is developed from the combination of both corresponding shapes and the contours of the initial shape. If no corresponding shapes are found in the design space, a representation is generated from both the selected shape and the leftover of the initial shape. Using this approach, a set of multiple representations can be developed through multiple selections of different shapes of interest.

(b) *To learn about shape semantics in relation to their situations. Constructing the situatedness of shape semantics through learning the regularities of the relationships among them cross the observations constructed from the design environment.*

This objective can be achieved by recognising shape semantics at each representation and constructing a set of observations for the corresponding representations. Shape semantics recognition starts from the identification of shape congruency and structural shape similarities (Cha, 1998). Each shape semantic has preconditions, necessary and sufficient, without which it will not be recognised in a design space. An observation is constructed from the group of shape semantics recognised from a representation of the design composition. The regularities of relationships among shape semantics across a set of observations help in constructing the situatedness of these shape semantics and locate them in relation to their situations in the design environment.

(c) *To implement a computational system that has the capability to: generate multiple representations; recognise shape semantics from the representations; construct observations; construct the situatedness of recognised shape semantics; and dynamically change what has been learned in response to the changes in the environment.*

This can be achieved by developing a computational system for situated learning in design called SLiDe. Its role is to locate design knowledge (shape semantics), in relation to its design situation by learning the regularities of relevant relationships among the shape semantics across observations from the environment, ie. learning the applicability conditions (situatedness), of design knowledge. SLiDe's framework consists of three modules: Generator, Recogniser and Incremental Situator that includes two sub modules: Situator and Restructuring Situator. The Generator module assists the designer to generate multiple representations of the initial design composition in which these representations serve as a platform and form the design environment for the Recogniser module. The Recogniser module detects shape

semantics in each representation in the design environment. The results of using the Recogniser module are sets of observations. Each observation comprises a group of shape semantics associated with each corresponding representation. The regularities of relationships among shape semantics across observations are the triggers for the Situator module to construct situational categories. These categories are dynamically updated using the Restructuring Situator in response to any changes taking place in the design environment.

The results of SLiDe are sets of situational categories that carry the applicability conditions of shape semantics. They are called situational categories to differentiate them from the normal categories found in unsupervised clustering mechanisms in a knowledge driven approach in machine learning (Mitchell, 1997). In unsupervised learning what the focus is and what the situation is, are not clear (Gero, 1998b). All regularities in a state description of a particular state of designing are candidates for both knowledge in focus and the situation. Within each regularity, a certain shape semantic could be the knowledge in focus and all the remaining shape semantics within this regularity become a candidate for the situation of that knowledge.

(d) *To explore ways of using SLiDe in designing architectural shape compositions.*

This can be achieved by demonstrating SLiDe's capabilities integrated with conventional CAD systems, during the designing of architectural shape compositions (SLiDe-CAAD), to provide interactive designing support at the conceptual stages of designing. SLiDe-CAAD could help designers, using conventional CAD systems or design sketches, in exploring the design space of their designs and allowing them to have a variety of representations of what has been designed. So that it may lead them to new moves from those they may otherwise have favoured. The representations of the same design composition could help to focus designers' attention to potentially hidden features of their design elements. Enhancing the perceptual interaction with design elements could be achieved by directing designers' attention to a set of shape semantics derivable from their current design by highlighting various sets of design elements that reflect these semantics so the designers might indicate which of these semantics is of interest. Having defined the designers' interest as the knowledge in focus, SLiDe-CAAD could dynamically change the association between design elements as the process of designing is developing. This could be achieved by maintaining the situation of that focus from the learned set of situational categories. Hence, SLiDe-CAAD could help to maintain the integrity of the designers' focus through preserving the relationships that define the situation of that focus. Using SLiDe-CAAD to provide such features to the conventional CAD systems has the potential to change the nature of these passive systems to be active and responsive designing systems at the early conceptual stages.

1.3 Scope and Limitations

The scope of this thesis includes knowledge driven situatedness of finding and locating regularities between structure and behaviour within a design composition. In carrying out

this research, it is acknowledged that the concept of situatedness involves different dimensions that can be taken into consideration while applying it to designing and learning. For instance, actions taken by designers that lead to specific situations within the view of achieving a certain goal could be traced back to learn the relationships between goal-oriented actions and encountered situations, ie goal driven situatedness. Within the proposed domain of architectural shape semantics, there are some other factors that can be taken into account when learning the situatedness of shape semantics such as the function of a space within a shape boundary. This may extend the focus from locating regularities between structure and behaviour to include the function. Such dimensions are beyond the scope of this thesis.

1.4 Organisation of the Thesis

The remainder of this thesis is organised as follows. Chapter 2 elaborates on the concept of situatedness of designing and learning, reviews the related work from cognitive science especially situated action, situated learning and machine learning in designing. Chapter 3 introduces the concept of multiple representations and its role in serving as a platform for a situated learning system. It presents how multiple representations can be generated from an initial representation of a design composition using infinite maximal lines and the process of selecting alternate shapes and searching for corresponding shapes to generate different representations. Chapter 4 addresses the recognition of shape semantics from the multiple representations, constructing sets of observations and learning the situatedness of recognised shape semantics. Chapter 5 introduces the framework of the computational model of situated learning in designing (SLiDe), its modules and its processes. Chapter 6 presents the results of using SLiDe within the domain of designing architectural shape compositions and its capacity to change its behaviour in response to changes in the environment. Chapter 7 demonstrates ways of using SLiDe in supporting architectural designing at the early conceptual stages such as enhancing the perceptual interaction between the designer and design elements and maintaining the integrity of the desired shape semantics. Chapter 8 outlines the contributions of this thesis and possible future research based on the results of this thesis.

Chapter 2

Background

"It was a long time before I fully realised the importance, for many psychological experiments, of putting the situations which are used to produce responses into sequential form" (Bartlett, 1958)

"Knowledge can only be created dynamically in time" (Newell, 1982)

This chapter reviews the research areas that have contributed to the research presented in this thesis. Different approaches to perceiving designing (rationality vs. reflection-in-action), and ways in which designers engage in designing (planned vs. situated), are discussed. The notion of situated action and cognition and how designing is viewed as situated are introduced. What is the situation and how it is constructed are defined. Situated learning as a general theory of knowledge acquisition where learning is related to the circumstances of its acquisition and its connection to the theory of situated action is addressed. Context-sensitive learning within the approach of situated learning is reviewed. Learning in designing and machine learning systems in designing are surveyed. The implications of situated learning for learning in designing are introduced and some works related to situated learning in designing are discussed.

2.1 Designing and Situatedness

2.1.1 Designing: rationality vs. reflection-in-action

Designing is taken to be a complex process that includes activities and tasks. The vast majority of views of designing are that it is an activity. It involves distinguishable processes that occur over time. Many systems of describing these processes of designing have been developed. The first generation methods of designing methodology were heavily influenced by the theories of technical systems. The thrust of these methods made designing to be seen as a rational process. It means staying within the positivist framework of science, such as physics, as the model for a science of designing. Simon (1996) quotes optimisation theory as a prime example of what he believes a science of designing could and should be. The problem solving approach prescribes looking at designing as a search process, in which the scope of the steps taken towards a solution is limited by the information processing capacity of the acting subject. The state space of possible designs is defined in advance and is bounded.

Describing designing as a rational system is particularly apt in cases where the problem is determined and well structured and designers have complete strategies that can be followed while solving them.

On the other hand, Schön (1983) criticises technical rationality (the basis of mainstream design methodology), arguing that design methodologists that work within this view restrict themselves to terms of generalities about the processes of designing. In Schön's opinion, too little attention is paid to the structure of the tasks of designing and the crucial problem of linking process and task in design situations. Schön proposes an alternative view of designing, based on the idea that "a kind of knowing is inherent in intelligent action". In "action oriented" designing, knowledge cannot be described within the prevalent methodology of technical rationality. The basic elements of activities are actions, and the kernel of the designing ability is to make intelligent decisions about those actions. Designers react to the new state of their own making. The final design is a result of this interaction. In the reflective conversation with the situation, designers name the relevant factors, frame a problem in a certain way, make moves towards a solution and evaluate those moves. During the naming-activity designers explicitly identify relevant objects in the designing task. The framing activity is to distinguish the context in which other activities occur. The moving activity is not only a state of trying to solve the problem but exploring the suitability of the frame. Reflection is a conscious and rational action that can lead to reframing the problem, when the frame is not satisfactory, making new moves or attending new issues (Valkenburg and Dorst, 1998). Describing designing as a process of reflection in action works particularly well in the conceptual stages of the designing process, where the designer has no complete strategies to follow in proposing and trying out problem/solution structures (Dorst and Dijkhuis, 1996).

2.1.2 Designing actions: planned vs. situated

Which comes first, the action or the plan? Some might say, the plan. According to Suchman (1987), saying this is not an appropriate way of understanding what really happens when a person sets out to do something. It is only when we have to account for our actions that we fit them into the framework of a plan. Actions are to a great extent linked to the specific situation at hand and are therefore hard to predict by generic rules. Action, learning, understanding and remembering is situated. Most of Artificial Intelligence related action research has assumed that the plan has an existence prior to and independent of the action. Intentions are viewed as the uncompleted part of the plan. This assumption does not account for intentions that are never realised. Designing has been viewed as intentional action (Galle, 1999), in which no plan was completely formed in advance. According to Suchman (1987), plans are representations of situated actions. The objectivity of the situations of our actions is achieved through the indexicality of language. By saying that language is indexical, indicates that the meanings in its expressions are conditional on the situation. Language is a form of situated action. Humans associate a word with a variety of different contexts, each of which contain one or more salient features that could trigger the use of that word. These features associated with a word are not just a list to be applied as they arise serially. Rather they are correlated in certain ways, and these correlations are important in applying the word. The situatedness of a word is the correlation of various features

associated with each other (Gee, 1997). As a consequence of the indexicality of language, mutual intelligibility is achieved on each occasion of interaction with reference to situation particulars, rather than being established once and for all by a stable body of shared meaning. Suchman (1987) concluded that building interactive tools has much to gain from understanding the context of situated human action.

2.1.3 The situated view of cognition/action

Situated cognition is the study of how human knowledge develops as a means of coordinating activity within the activity itself. Situated cognition acknowledges that every human thought and action is adapted to the environment, that is, situated, because what people perceive, how they conceive of their activities, and what they do develop together. This means that feedback occurring internally and within the environment over time is of paramount importance. Therefore, knowledge has a dynamic aspect in both format and content. 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, 1997b) acknowledges that "where you are when you do what you do matters". A central tenet of the situated action approach is that the structuring of activity is not something that precedes it but can only grow directly out of the immediacy of the situation (Suchman, 1987; Lave, 1988). Every activity is by definition uniquely constituted by the confluence of the particular factors that come together to form one situation (Nardi, 1996). The implication of this view is that actions are interrelated to their locus and application. Situatedness or context dependence is seen as a general principle of human knowledge and activities, and cognition is a constructive process taking place in a situated background (Rappaport, 1998). It is a shift in perspective from knowledge as a stored artefact to knowledge as a constructed capability-in action. This shift is in contrast to the rational approach in which the theory of situated cognition acknowledges that when modellers equate human knowledge with a set of descriptions (a collection of facts and rules in expert systems), they are describing abstractly how the program should behave in particular situations. They are not capturing the full flexibility of how perception, action and learning are related (Clancey, 1997b).

Situativity theory is used as a replacement term for situated cognition by Greeno (1995), leading exponent of situated cognition, who expressed dissatisfaction with the term situated cognition claiming that all cognition is situated. But Bereiter (1997) asserts that there is a non-situated cognition and situativity theorists have devoted a lot of effort to criticising it. Non-situated cognition cannot be found in humans. It can be found in machines. Most artificial intelligence has been constructed to a model radically at variance from the model suggested by situativity theory. In situated cognition, people or other agents carry on activities adapted to the environment. Machine intelligence of the classic AI kind is not of that type in which cognition is entirely a process of symbol manipulation (Vera and Simon, 1993). Interaction with the outside world is done by means of transducers that translate inputs from sensors into symbols that the machine can manipulate or translate symbols into actions. One very important line of argument in favour of situated cognition comes from roboticists, who find that the non-situated cognition does not work very well (Beer, 1990).

If we take rule-based expert systems as exemplifying non-situated cognition, then looking at what they may offer would give us some insights into situated cognition. Rule-based expert systems work very well when all the necessary information can be explicitly represented and indexed, as in the case with a game of chess. The rule-based expert system can then work on its stored information and produce results to the part of the real situation of which it contains representations; for example, it can compute a move appropriate to a real chess game. The trouble is that the great bulk of real world situations cannot be represented in this way, simply because they are unknown until they are encountered and experienced. Although non-situated cognition may not be very good for capturing the situatedness of activities, it has proved to be capable of guiding a space vehicle to Mars (Bereiter, 1997).

Saying that the cognition is situated means that reasoning processes are not merely conditional on the environment, but are inherently brought into being during an interactive process. Information in the environment is not merely described, selected, or filtered, but constructed in the course of perception. Categorisation of things in the world are not merely retrieved descriptions, but created new each time (Clancey, 1991). From a psychologists' perspective, the theory of situated action (Mills, 1940; Suchman, 1987; Thelen and Smith, 1994) acknowledges that knowledge is dynamically constructed as we conceive of what is happening to us in our moves and the activity that we are currently engaged within. A dynamic adaptation is always generalising our perceptions, our actions and coordinations as we act. This reconceptualisation occurs moment by moment and is a necessary part of abandoning a plan and looking more carefully to recategorise the situation. Situated cognition acknowledges that we are always automatically adjusting even as we follow a plan. We are always recategorising circumstances, even though we appear to proceed in locked step with our prescribed actions (Clancey, 1997a).

2.1.4 Designing as a situated activity

In designing there are things to know, ways of knowing them and finding out about them. The designerly way of knowing has been identified in an attempt to understand how designers work. It is suggested (Baker, 1993) that designers share a solution focused strategy, which allows them to learn about a particular problem by generating a set of possible solutions to it. This is different from the more scientific definition of a solution as the result of a process of optimisation or formal analysis. Each possible solution is a different perspective (representation), of how the designers are looking at the problem. In this thesis, designing is viewed as a situated activity in which designers interact with their design environment and bring their prior experience to the particular situation. The way in which designers interact with objects in the design environment and find out about these objects is based in part on what is available in front of them at that moment of time in the environment and their prior experiences brought to the situation. Their interactions with the design environment cannot be completely planned in advance, simply because designers do not know in advance what will be available in the design environment in order to pre-plan their actions. Designers’ actions take place in situations. The effect of this, is that designing is an activity that does not exist except in relation to situations and design knowledge cannot be fully understood or explained in isolation from its situation.

Design problems are described as ill-structured because one never has sufficient information in the initial state when the properties of the goal state are not fully specifiable in advance; and therefore many different goal states are conceivable (Goldschmidt, 1997). The implications of various design actions are not always predictable. For instance, if a new object is added to a design, it is not possible to fully predict that object's impact on the design. This new object might combine with other objects, or might not interact with other objects at all (Brown and Birmingham, 1997). This clearly indicates that it is not possible for designers to know beforehand what particular set of states they would encounter; and in consequence of what kinds of actions they might take and similarly what kind of results they might achieve. This is simply because they travel among different surroundings in the environment in relation to the goal state in which these relations might change and their effect might lead to different situations.

Thus, it is claimed that designing is a process experienced within the situation encountered by designers. This coincides with the most recent view from cognitive science that has started to provide some insight into human activities where cognition and knowledge are emergent properties of the interaction of an individual with the environment, ie. the current situation (Clancey, 1997b; Clancey, 1998). Accounting for the situation entails that learning methodology cannot be limited to the task at hand but has to take into account the whole environment in which the task has to be preformed.

2.1.4.1 Situatedness in designing

In conceptual designing, designers work with their experiences, their knowledge and their perception of what is in front of them in order to determine what may be described more formally as the variables that contribute to the function, behaviour and the structure of the resulting design. The particular behaviour and structure variables are not only chosen a priori but are produced in response to the various situations as encountered by designers. What the designer has done previously, both prior to this design and during the current process of designing, affects how designers view the situation and what memories they construct and bring to bear on the current situation (Gero, 1998a). For instance, Figure 2.1 demonstrates the concept of situatedness through an example of emergence and shows graphically how the situation can affect what is to be perceived. Figures 2.1(a) and (b) show only one head at some time and designers may never emerge a vase as they may do with Figure 2.1(c) and may never use that emergent figure in later designing. It is only when the situation exists where both heads appear at the same time, Figure 2.1(c), that the emergent vase appears. With no emergent vase it is not possible for designers to be influenced by it or to change their design trajectory on the basis of it. Furthermore, if the designer were to choose to reflect the image in Figure 2.1(c) around one of its vertical edges as it can be seen in Figure 2.1(d), it is only then in this situation that not only a new emergent shape (decorative arch) appears in the middle between the two vases, but also an emergence of a new relationship, that is: reflective symmetry of the two vases around a vertical axe.

(a) (b) (c) (d)

Figure 2.1 (a) and (b) show only single dark human-like heads at some time; (c) it is only when the situation exists when both heads appear at the same time facing each other, that the emergent vase appears; and at (a) and (b) different situations have arisen and therefore no emergent vase can be found; (d) shows a new situation where the designer chooses to reflect the image in (c), it is only then that a new emergent shape (decorative arch), appears in the middle between the two vases; in addition to the emergence of a reflective symmetry relationship between the two vases (after Gero, 1998).

The concept of situatedness is not necessarily tied to any particular representation such as the graphical example demonstrated in Figure 2.1. However, each representation has the potential to provide different situations and as a consequence different interpretations of what the situation is. It provides a basis for the delineation of the indeterminism of designing. Situatedness can be seen as a means by which the designer changes the trajectory of the developing design. Different situations provide different opportunities to move in different directions considering that neither the situation nor what is being focused on is given but is a function of the designer's interpretation of what is there and how the situation is constructed. This may explain in part why designing is not a predictable activity and provides insight into why designing often leads to unexpected discoveries. Schön (1987) described the concept of situatedness in designing briefly as: "He shapes the situation ... his own methods and appreciations are also shaped by the situation". There is increasing evidentiary support for situatedness in designing (Goldschmidt, 1997; Lueg and Pfeifer, 1997; Suwa et al., 1998b; Lueg, 1999).

2.1.4.2 Situated versus procedural and declarative knowledge

Declarative knowledge describes how things are. This is accomplished through the description of objects (office, building, and entrance), their attributes, (functional, open and attractive), and the relations between them (functional building, open entrance and attractive office). Procedural knowledge describes and predicts actions or plans of action. All knowledge of “how-to” (How to make stairs? How to construct a building?), are examples of procedural knowledge. Before the declarative knowledge can be of use, an understanding of how the goal state is linked to the initial problem state must be present and a set of transformations to accomplish this must be developed, ie procedural knowledge.

Situated knowledge is captured and associated with a specific situation. Situated knowledge about an object would be understood as the relationships between this object and a social or physical situation rather than simply a property of the object. In other words, it is these relationships that help to locate a certain piece of knowledge in its specific situation within which it was recognised. A relativised concept of situated

knowledge would be analogous to the concept of motion in physics. The velocity and acceleration of an object in motion are not properties of the object itself, but are properties of a relationship between the object and a frame of reference (Greeno, 1989). Within this view, knowledge could be seen as situated.

2.2 What is the "situation" and how it is constructed?

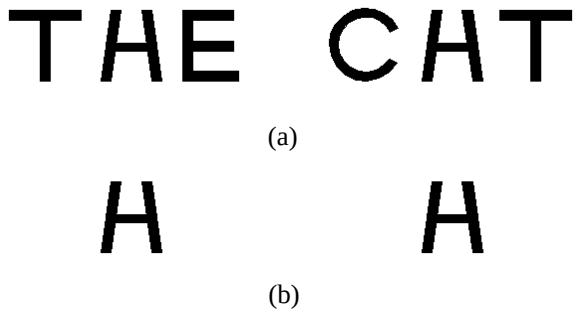
Is a situation a moment of time? Is it a location? Is it a life situation, a social situation, or a configuration of relationships? Or is it perhaps more like a position, a perspective, and a viewpoint of the subject? All of these aspects play a variety of roles in the discourse on situated cognition. Situatedness is not a black box; it is more than an open box that offers a rich variety of interpretations and possibilities (Engestrom and Cole, 1997). Gee (1997) states that "the word situated itself takes on a somewhat differently situated meaning".

Situatedness has antecedents in the work of Heidegger (1927), Bartlett (1934) and Dewey (1939). Heidegger (Heidegger, 1927; Stahl, 1993) defines the situation as the person's sensitive context including the physical surroundings, the available tools, and the circumstances surrounding the task at hand within the person's aim. Bartlett gave a very broad definition of what constitutes a situation. He claimed that a situation cannot be adequately described merely as a series of reactions, or merely as an arrangement of sensations, images, ideas or train of reasoning. A situation always involves the arrangement of cognitive materials by some more or less specific active tendency, or groups of tendencies. To define a situation in any given case we have to refer not only to the arrangement of material, but also to the particular activities in operation. On the same line of thought Dewey (1939) emphasised that a situation is not a single object or event. A situation refers to our experiencing objects and events in connection to a contextual whole.

In the situated view of design knowledge, the situation is defined as the relevant context from the environment in relation to a specific aim or focus and is constructed. This relevant context is relevant to that focus where other contexts in the environment are irrelevant. The relevancy of the context in relation to the knowledge in focus is determined from the regularities constructed from the environment. Thus, a situation is not simply the context in which the context is conceived very broadly. For instance, the physical surroundings of visual objects have an impact on basic perception (Solso, 1997). Consider Figure 2.2, where the same stimulus  produces different perceptions depending on the situation within which it is perceived. Figure 2.2(a), when read is "THE CAT"; yet upon close inspection, the stimulus  in "THE" is the same as in "CAT". If H/A were presented in isolation as in Figure 2.2(b), out of context, we would be confused as to their correct identity. The identification of each is based upon the situation.

Borrowing from Barwise and Perry (1983) and Barwise and Seligman (1997) assert the idea that a situation is determined by "uniformities" (regularities), and composed of a collection of entities. Each entity may have a set of properties associated with it. Furthermore, there are some relationships of the entities to one another in the situation.

Typically, these can be described as relationships that support 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 relationships within a situation are the meaningful associations among entities that provide structure to the situation. The regularities of relationships among entities give rise to a learning system to encounter the situation. Such a system should be attuned to these regularities.



(a)

(b)

Figure 2.2 An example of situation. The same stimulus  is perceived as an H or an A depending on the situation (after Solso, 1996).

Examples of the situatedness of a table in an office space or in a dining room are described as follows. The relationships between a table and other entities in an office space are regularities across observations recognised in its environment. In a dining room, there is another type of relationships between the table, even though it may be the same object, and other entities. These regularities of relationships are not natural laws nor causal effects, but rather automatic consequences of differentiating the relationships in the first place. It is these relationships that allow the situated knowledge about a desk and a dining table to be constructed in relation to their situations from the environment within which they were recognised. Within this comparison, we may make a distinction between what we mean by context and situation. In an office space there might be many objects (entities), that surround a table, such as drawers, filing cabinets, computers, desk lamp, walls, windows, etc. These objects constitute the whole context of a table. The situation of that table that made it to be recognised as a desk is comprised of the salient features from the surroundings which are drawers, filing cabinets, a computer and a desk lamp. In an open office space, walls and windows are not salient features of the surroundings of that table, however the table is still to be recognised as a desk. So, the situation is the only relevant part of the context in relation to the focus.

Following Barwise and Perry (1983), a distinction is made between 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 and the same relation to one another. In contrast, a dynamic situation is a course of events composed of a series of frames of related events that are linked through some common entity, ie regularity. For instance, a mail carrier making a phone call in which the entities, their properties and their relationships remain the same is a static situation. In contrast, the situation of the mail carrier delivering the mail in a route is a dynamic situation whereby different entities such as different houses, streets,

scenes, obstacles, etc. and different relationships are involved (Radvansky and Zacks, 1997).

One of the main characteristics of designing is its dynamic nature. During the process of designing, both knowledge and situations are not 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-relationship structures, which may lead to modifying previous situations or creating 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. In this thesis, the role of situatedness is to locate design knowledge in its situation by learning the regularities of design knowledge relationships across different observations from the environment within which this knowledge was recognised.

The following is a simple analogy to constructing and learning situations. Imagine that you are reading a particularly engaging detective story 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 relationships 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).

2.3 Situated Learning

Situated learning is a general theory of knowledge acquisition where learning is seen to be more closely linked to the circumstances of its acquisition than previously acknowledged (Lave and Wenger, 1991; Billett, 1996). Situated learning addresses the relatedness of actions to situations and to the bringing forward of the contextual embeddedness of learning (Lave and Wenger, 1991; Kirshner and Whitson, 1997). It is based on the concept that knowledge is contextually situated and is fundamentally influenced by the context in which it was acquired. The nature of the situation and circumstances in which knowledge is appropriated is influential in determining the likely prospect of subsequent redeployment to other situations (Billett, 1996). Lave and Wenger (1991) argue that learning as it normally occurs is a function of activity and the context in which it occurs, ie it is situated. This contrasts with most learning systems which involve knowledge that is often presented out of context, ie context free. Furthermore, situated learning may be incidental rather than deliberate. Brown et al (1989) argue that knowledge is situated, ie context dependent, and that activity and

situations are integral to cognition and learning. Suchman (1987) claimed that learning entails a form of context-bound and embodied situational action. Every course of action depends in essential ways upon its material, and circumstances. So, learning is not simply a matter of ingesting externally defined, uncontextualised objects, but a matter of developing context-bound discourse-practices. Learning knowledge in a relevant context motivates learning and ensures that it is usable (Streibel, 1995).

What does it mean to say that learning is a process of knowledge construction that is highly tuned to the situations within which it takes place? Two interpretations, at least, are important: The neurophysiological view is that perception and action arise together, so one's knowledge is always a new way of coordinating ways of talking, seeing and moving within on-going interactions (Clancey, 1993). In this sense, learning is tuned to situations because our perception of what constitutes a situation is arising within a newly organised and adapted response. The social view is that the use of tools occurs within social interactions, so that the idea of task is enlarged to "participating as a member of a community of practice" (Lave and Wenger, 1991).

Context-sensitive learning as related to situated learning

Many practical learning applications necessitate the use of context in learning (Matwin and Kubat, 1996). Activity theory (Nardi, 1996) proposed a specific notion of the context in which context is related to some extent to the definition of the situation as adapted in this thesis. 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). A variety of means of studying context conditioning has been developed. Some procedures examine the effects of context in response to specific features embedded in that context, whereas other procedures examine behaviour controlled by the context itself. A context is constructed every time a system is activated in a setting: finding out what is available and examining prior experience. Context, then, is the result of this process of identification in a particular activity setting. Within this view, context is never universally given, nor objectively predetermined (Ores, 1998).

Turney (1996) argued that learning systems that can identify the context would have a substantial advantage over a system that require a human to identify the sensitivity of context. An understanding of surrounding events and accompanying dialogue is vital if our utterances are to mean what we want them to mean. So too in designing it is necessary to take account of the context within which an artefact and its various components and subcomponents are to function. Coyne and Gero (1985) reformulated the design rules so that they are sensitive to context and define contextual attributes as those that appear on both sides of the transformation rule. They describe a transformation rule belonging to a context sensitive design grammar as one that operates on a particular pattern in the state description by transforming it into another pattern, provided that other patterns are present. Those other patterns constitute the context for that rule and are not changed by the rule. They must be present if the rule is to be fired.

2.4 Machine Learning in Designing

A popular definition or description of the process of designing is as a goal-oriented problem-solving activity (Archer, 1965). The designing process has been described as the cycle of design analysis, design synthesis, and design evaluation (Asimov, 1962; Jones, 1963; Dasgupta, 1989). In this model of designing, well-structured knowledge is needed. Design situations, design process, and design decisions are predefined and described in some symbolic representation. In this view of designing, the relevance of all designing activities is fixed beforehand; consequences can be intended with no need to reflect on design actions. Based on this metaphor, design is an action within an assembly of symbols, patterns, and planned sequences (Sun, 1993). Based on this view designing has been modelled as search (Coyne et al., 1990; Russell and Norvig, 1995) within a given representation. Designing has recently been modelled as a form of exploration (Logan and Smithers, 1993; Gero, 1994), where the design space that is to be searched has first to be constructed or located. Both these views are founded on the notion that knowledge exists outside of its use and only has to be applied to be useful. Thus, machine learning in designing is concerned with finding relationships between structure and behaviour and representing that as knowledge to be applied later.

Each design that a designer works on adds to the experience of the designer. In this sense, the designer learns from each design (Gero, 1996). Designers learn when they encounter knowledge that is sufficiently different from their present state of knowledge (Persidis and Duffy, 1991; Duffy and Duffy, 1996a; Duffy, 1997). In other words, memory provides the foundation upon which learning takes place. Towards understanding learning in designing, Persidis and Duffy (1991) and Duffy (1997) discussed how learning in designing occurs, what knowledge is learned, and when learning occurs? To understand learning it is useful to examine what constitutes a learning event. Learning might occur in three basic ways: acquisition, generation and modification. Acquisition represents the process of receiving new knowledge, generation represents the process of creating new knowledge from existing knowledge and modification represents the process of altering existing knowledge. There are no clear boundaries to the design knowledge to learn, however there are main areas. For instance, we may learn from the environment in which the design solution operates; the design description of the solution; and the domain in which the design solution belongs. Learning is a perpetual process that occurs both during and within designing.

Learning and designing are closely related activities. Identifying design problems involves the use of knowledge learned from previous design solutions. Searching for an alternative design solution can also be somehow guided by the knowledge learned from a previous design failure. The evaluation of a design solution can usually be based on the knowledge learned from generalised features of the proposed design solutions (Smithers et al., 1993). Learning activities can take several forms and assume several roles learned in the designing process (Reich et al., 1993): learning technical knowledge; learning about the problem, its solution, and their relationships; and assimilating experiences for future use.

Learning systems in designing

The ability to learn must be part of any system that would claim to possess intelligence. The original objective of machine learning has always been the automated generation of knowledge in different forms such as decision rules or trees (Reich, 1997; Reich, 1998). Intelligent systems constantly adjust their knowledge and expectations in the course of their interactions with their environment, as well as through the experience of their own internal states. The dynamic nature of knowledge plays an important part in the development of computer-based designing systems. The mainstream of computer-aided designing systems has focused on the formulation of solutions rather than the generation and modification of knowledge used to create these solutions. This applies to the development of artificial intelligence in designing systems and intelligent CAD systems (Duffy, 1997).

Machine learning is a branch of AI concerned with the study and computer modelling of learning processes. Machine learning is primarily composed of (Carbonell, 1990; Langley, 1996): inductive learning, eg acquiring concepts from positive and negative examples; analytical learning, eg explanation-based learning and certain forms of analogical and case-based learning methods; genetic algorithms, eg classifier systems; and connectionist learning methods. Machine learning techniques can be classified according to the level (knowledge, symbol, or device) at which knowledge representations (such as, rules, frames, predicate logic, semantic networks, classifiers, conceptual clustering and genetic algorithms) can be expressed (Kocabas, 1991). The main machine learning techniques applied in designing include (Duffy, 1997): agent based learning, analogical learning, case-based reasoning, induction, genetic algorithms, knowledge compilation and neural network systems.

The development and applications of machine learning in designing did not receive much attention until the late 1980s. An overview of machine learning systems in designing extracted from an extensive review (Duffy, 1997; Sim and Duffy, 1998) is illustrated in Tables 2.1(a) to (c) summarising how learning systems in designing assist in the utilisation of experiential design knowledge. In this overview, six main elements of the learning process have been presented: input knowledge ($\mathbf{I}_k$), goal or reason for learning ($\mathbf{G}_l$), knowledge transformers ($\mathbf{K}_t$), output knowledge ($\mathbf{O}_k$), what triggers learning ($\mathbf{T}_{tw}$) and when learning is triggered ($\mathbf{T}_{lt}$); in addition to two other elements: the way in which design knowledge is represented and the methods of machine learning that were used. The elements of concern here are what triggers the learning and when that trigger is likely to occur. Knowing them is important if machine learning capability is to be incorporated into designing support systems. It is also important to relate them to the context of the knowledge learned and the knowledge transformers involved showing the relationships between these elements of learning (Sim and Duffy, 1998). Tables 2.1(a) to (c) show various learning systems in designing in which learning design knowledge takes place under different types of triggers: retrospective, *in situ* and provisional. Retrospective triggers for learning design knowledge can occur at the end of the designing process while *in situ* triggers of learning occurs during the designing process, while design decisions are being made. These decisions may lead to successful design action or a failure. An example of *in situ* trigger is the violation of design expectations (Chabot and Brown, 1994).

Table 2.1 (a) Overview of learning systems in designing.

Machine learning in design	Input I_k	Goal G_l	Knowledge Transformer K_t	Output O_k	What triggers learning, T_{tw}	When is learning triggered, T_{lt}	Design knowledge represented	Machine learning methods
(McLaughlin and Gero, 1987)	Past design configuration and performance evaluation criteria	Excellence driven to achieve better design	Group rationalisation	Clusters of design configuration map to performance evaluation space	Performance trends in new design	Retrospective	Clusters of archetypes of design solution mapped to performance evaluation space	ID3/Pareto
BOGART (Mostow, 1989; Mostow et al., 1992)	Several plan instances	Change the design plan by reasoning from previous plans	Similarity/ dissimilarity comparison	Knowledge of design plan.	New but similar design	In *Situ*	Design plan as a hierarchal goal structure	Derivational analogy method
			Generalisation	Generalised design rules Generalised design plans	No existing design rules Module(s) in plan refined	In *Situ* In *Situ*/ retrospective	Generalised design rules Generalised design plans	LEAP using EBL generalisation VEXED's ability
ARGO (Huhns and Acosta, 1992)	Records of design action described by preconditions and post-conditions.	Streamline design process by replaying a similar plan	Abstraction	Abstracted design plan by removing leaf nodes from plan	New abstract design concept	Retrospective	Abstract plan of macro-rules	Merging edge macro-rules into cumulative macro by removing leaf rules
DONTE (Tong, 1992)	Set of sub-problems that are presumed to be independent	Learn design control knowledge to optimally search the design space	Explanation/ discovery	Search control knowledge	Optimal design solution	Provisional	Discovery of macro-decision rule to reduce search	Hypothesis formation Hypothesis test
BRIDGER (Reich, 1993)	Instances of past designs together with existing taxonomic knowledge	Expedite synthesis of preliminary design concepts	Group rationalisation	Taxonomic knowledge of design concept	New concept	Retrospective	Hierarchal structure of concept/sub concept	ECOBWEB/ EPROTOS

Table 2.1 (b) Overview of learning systems in designing.

Machine learning in design	Input I_k	Goal G_l	Knowledge Transformer K_t	Output O_k	What triggers learning, T_{tw}	When is learning triggered, T_{lt}	Design knowledge represented	Machine learning methods
CONCEPTOR (Li, 1994)	Instances of past designs in terms of attributes and attribute values	Expedite preliminary definition of form and structure of design concept.	Group rationalisation	Empirical knowledge of quantitative information Design patterns of qualitative relationships	Performance trends in new design	Retrospective	Decision tree of rules for Pareto optimum design	COBWEB
			Derivation / randomisation		New/updating empirical relationship(s) New/updating design patterns	Retrospective	Empirical formula Design patterns	Concept aggregation
DIDS (Wang and Howard, 1994)	Knowledge of failed constraints	Streamline design process by detecting and avoiding design failure	Similarity/ dissimilarity comparison	Knowledge of anticipated crucial constraints	New design case	In *Situ*	Design plan / history	Case-based reasoning
	Records of design action described by preconditions and post-conditions.	Streamline design process by replaying a similar plan	Abstraction	Abstract from session control knowledge of : Global design plan Related redesign plan Constraint violations	Past design cases to improve design process of similar designs Crucial constraints that triggered redesign process	Retrospective Provisional	A global design plan and several design plans	Identify and classify all knowledge source activation records
DSPL (Chabot and Brown, 1994)	Knowledge of current design constraints	Streamline design process by detecting and avoiding design failure	Specialisation	Specialised design constraint knowledge	Constraint violation	In *Situ*	Generalised design plan. Constraint rules.	Knowledge compilation through constraint inheritance
IDEAL (Bhatta and Geol, 1994)	Past designs (structure-behaviour-function)	Learning physical principles of a concept without knowing the target concept a priori	Explanation/ discovery	Model of physical principles	Functional-driven design	Retrospective	Discovery of physical principles from abstract design models	Hypothesis formation Hypothesis test

Table 2.1 (c) Overview of learning systems in designing.

Machine learning in design	Input I_k	Goal G_l	Knowledge Transformer K_t	Output O_k	What triggers learning, T_{tw}	When is learning triggered, T_{lt}	Design knowledge represented	Machine learning methods
NETSYN (Ivezic and Garrett, 1994)	Past designs to train the neural network to estimate the desired probabilities	Learn the Bayesian a posteriori probabilities of design properties.	Derivation / randomisation	Posterior probabilities of design properties.	New knowledge of a posterior probabilities	Retrospective	Bayesian a posterior probabilities of design properties represented as network of weights in neural network structure	Modular back propagation neural network
NODES (Duffy et al., 1995)	Instances of past designs together with compositional knowledge, eg. part-of and kind-of	Expedite preliminary definition of form and structure of design concept.	Association/ disassociation	Compositional knowledge of design concept	New design configuration	In *Situ*	Compositional network of concepts Numerical network of concepts	Semantic links in network
		Enrich its design knowledge –base	Generalisation	Generalised design concept	New concept saved	In *Situ*	Generalised rules of design concepts	Maximal Conjunctive Generalisation
CDA (Britt and Glagowski, 1996)	Past of working designs and predefined domain rules	Detailing to reconstruct design history	Detailing	Detailed design plan reconstructed bottom-up	No similar design plan existed	Provisional	Detailed design plan built bottom-up	Reconstructive derivational analogy
(Murdoch and Ball, 1996)	Past design configuration and performance evaluation criteria	Excellence driven to achieve better design	Group rationalisation	Clusters of design configuration map to performance evaluation space	Performance trends in new design	Retrospective	Clusters of archetypes of design solution mapped to performance evaluation space	Kohonen neural network/ GA
PERSPECT (Duffy and Duffy, 1996)	Past design concepts described by attributes and values	Excellence driven by utilising knowledge from multiple sources	Abstraction	Multiple forms of explicit/implicit design knowledge	Non-existence of useful empirical equation or insufficient knowledge of attribute values	In *Situ*/ provisional	Abstracted empirical equation	ECOBWEB/DESIGNER

The concept of customised viewpoints (Duffy and Kerr, 1993; Duffy and Duffy, 1996b) is another example of learning design knowledge *in situ*. Depending on the design perspective that best suits the designer's current problem solving situation, PERSPECT (Duffy and Duffy, 1996b) can generate *in situ* multiple forms of implicit experiential knowledge through generalisations of past designs information. The most suitable generalisation of past designs that supports the current customised viewpoints for the design is identified. Provisional triggers such as control knowledge, DONTE (Tong, 1992), to search the design space is learned provisionally in anticipation of reducing the complexity of the search; or failed constraints, DIDS (Wang and Howard, 1994), anticipated in redesigning. On the other hand, Grecu and Brown (1996) identify some other reasons for learning such as: novelty driven, excellence driven and failure avoidance driven.

Reich (1997) introduced the "timing" dimension as one of the key dimensions of learning systems. The timing dimension ranges from: *early* in systems that learn pro-actively by receiving data, learning and storing knowledge that can subsequently be used for problem solving; or *late* in systems that learn reactively by storing data and subsequently retrieve it, learn locally from it and adapt it to solve problems. Most machine learning systems learn pro-actively, such as ECOBWEB (Reich and Fenves, 1992), except for instance based learning systems (Nicolas Lachiche and Marquis, 1998) or case-based reasoning (Kolodner, 1993; Maher et al., 1995; Lenz, 1998) that are reactive systems.

2.5 Situated Learning in Designing

Machine learning paradigms are used within a shared view of the role of machine learning in designing: 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). There are benefits in having the tools unchanged by their use as this makes them independent of their use and they can be used with any arbitrary problem. However, there are significant disadvantages in having tools unchanged by their use. Each design that a designer works on adds to the experience of the designer, in this sense the designer learns from each design. 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 the knowledge in relation to its situatedness.

The dimensions of machine learning in designing have been developed further since primarily outlined by Presidis and Duffy (1991). Grecu and Brown (1996) introduced a set of dimensions for machine learning in designing inspired mainly by the attempts to apply machine learning in designing. Leading towards working systems, Reich (1998), built on previous work that dealt with the practical use of machine learning in designing (Reich et al., 1993; Reich, 1994; Reich, 1997) and developed a set of dimensions of

learning contexts as a top down analysis of learning in designing. Table 2.2 maps Presidis and Duffy (1991), Grecu and Brown (1996) and Reich (1998) dimensions of machine learning in designing reflecting the evolution and development of machine learning techniques in designing and can be seen as complementary to each other.

Table 2.2 Dimensions of machine learning in designing.

Reich (Reich, 1998)	Grecu and Brown (Grecu and Brown, 1996)	Presidis and Duffy (Presidis and Duffy, 1991)
Who is learning?		
Why does the learner want to learn?	What can trigger learning?	What can trigger learning?
When does the learner learn and when the results are needed?		When is learning triggered?
What is the learner doing?		
What is learned?	What might be learned?	What knowledge is learned?
How does the learner learn?	Elements supporting learning Availability of knowledge Methods of learning Local vs. global learning	How is learning carried out?
What are the consequences of learning?	Consequences of learning	
How many resources are needed to carry out the learning activity?		

Situatedness of learning in designing 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 designing deal with the environment independently from the situational conditions, ie context free systems. However there are a few systems that are related to some concepts of the situatedness as discussed in Sections 2.5.1 and 2.5.2. A situated learning system distinguishes knowledge that has been learned in multiple contexts by moving through these contexts or states of situatedness where this knowledge is elicited to situate it within its environment. The situatedness of knowledge carries with it aspects of the situation within which it was acquired. Supporting situatedness within a learning system is an initial attempt and important step towards building support systems in designing. In this sense, acknowledging the concept of situatedness is of importance to provide a system to have the capability to add another dimension to the existing learning systems in designing. The new dimension is to learn within which design knowledge was learned.

2.5.1 Incremental learning systems in designing

The situatedness of designing includes a set of concepts such as the dynamic nature of design knowledge and the relationships between knowledge and its locus and application. There are some machine learning systems in designing that are related to

the dynamic nature of learning design knowledge. Learning incrementally and detecting changes in an environment are important to knowing when it is time to learn. ECOBWEB (Reich and Fenves, 1992), is a system 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 recently another learning mechanism was introduced by Widmer and Kubat (Widmer and Kubat, 1996). Another two systems that are related to situated learning in the view of dynamic and incremental learning are discussed: BRIDGER (Reich, 1993) and PERSPECT (Duffy and Kerr, 1993; Kerr, 1993).

BRIDGER (Reich, 1993) is a domain independent learning system for knowledge acquisition and performance improvement built on the foundations of the concept formation program COBWEB (Fisher, 1987), but extended along several dimensions. BRIDGER's framework is based on an assumption that designing is a sequence of five tasks: problem analysis, synthesis, analysis, redesign and evaluation executed sequentially with one feedback loop. It is a system built for assisting in the conceptual designing of cable-stayed bridges. BRIDGER uses an incremental learning scheme for the creation of hierarchical classification tree and ECOBWEB is a major component of it. It partially implements the constructive induction mechanism (Reich, 1991) for the incremental concept formation. Constructive induction (Rendell, 1988) is defined, as the creation of useful features not existing in the original property-value description of examples. In constructive induction, collection of design examples are used by a learning system to produce a sequential collection of design rules, each representing different semantics. Within the view of situatedness, design knowledge constructed in BRIDGER is unrelated to its locus.

PERSPECT (Duffy and Kerr, 1993; Kerr, 1993) was originally developed to demonstrate and evaluate the design utility of customised viewpoints. The system is a designing tool that aims to support the experiential knowledge in numerical engineering design. Designers require different viewpoints from past design and abstractions in order to facilitate the effective utilisation of past designs to suit a designer's particular needs. PERSPECT is a dynamic design tool capable of automating the rationalisation of past designs to suit a designer's particular needs and support the automatic generation of customised design perspectives. The rationalisation of past designs is motivated by the belief that, within the abundant explicit information of individual past designs, there exists a wealth of implicit knowledge which should be made explicitly available to the designer.

Recently, Duffy and Duffy (1996b) utilised PERSPECT within the concept of controlled computational learning. PERSPECT is used within the concept of shared learning. PERSPECT presents how the activities of designing and learning are coupled. That is, where the designer, at various designing stages, develops a design solution, feeds such learned knowledge back to some store of experiential knowledge and reuses this knowledge to aid in the evolution of an acceptable design solution. During the development of a design solution from an initial stage to a design solution, some of the learned knowledge will transform to longer-term experiential knowledge and some will only be used in designing within the next stages of development. Thus, the experiential

knowledge reflects the knowledge that will be reused in later design scenarios, whereas the transient knowledge learned will only be used to assist in the evolution of the design to a final stage.

PERSPECT's functionality (Duffy and Duffy, 1996a) has been used to explore the possibility of realising the learning assistance in IDA (MacCallum et al., 1987), Intelligent Design Assistant. The collaboration between the two players of intelligent CAD (the designer and IDA), provides more effective learning capabilities. Shared learning requires more controlled computational learning to ensure that computers learn design knowledge that is relevant, useful, and understandable to designers based on their required knowledge needs. Controlled computational learning should not be confused with supervised learning. Controlled computational learning emphasises that designers should be able to manipulate computational learning such that the resulting knowledge mirrors what is required by designers to either assist the problem solving or to enhance their understanding/learning. Controlled computational learning has been proposed as one means of achieving shared learning. PERSPECT is related to situated learning in designing but is significantly different. PERSPECT attempts to reflect the dynamic nature of learning in designing through the generation of multiple and dynamic generalisations, to reflect and capture designers' changing interests.

2.5.2 Accommodating the situatedness within computational systems in designing

There are attempts to integrate some views of the concept of situatedness with learning systems that provide a dialogue environment for designers learning from prior designs such as Case-Based Reasoning (CBR) (Sooriamurthi and Leake, 1994; Oehlmann et al., 1995). CBR provides a reminding environment to assist designers to use past designs instead of designing from scratch. Since designer's actions depend upon the current situation, the automated adaptation of a design case is difficult. Because the situations vary over time, it is not possible to predefine the adaptation process of the system to involve unknown situations. However, the process of adaptation can be defined within a particular range of expected situations. A primary difference between such systems and the proposed situated learning approach is that design situations are not predefined but rather encountered, constructed and modified during the process of designing.

Another attempt such as Situated Design (Pfeifer and Rademarkers, 1991; Rademakers and Pfeifer, 1992; Hofmann et al., 1993; Müller and Pfeifer, 1997; Lueg, 1999) capitalises on the notion of humans as situated agents. Situated Design can be seen as the initialisation of a process of continuous learning and change. The focus is on support of learning, not rationalisation or automation. Situated Design accounts for situatedness and entails that the design methodology cannot be limited to the task at hand but has to take into account the environment in which the task has to be performed. Situated Design has been applied in several large cooperative ventures with industrial partners aiming at inducing change by enabling and enhancing communications (Müller and Pfeifer, 1997). Another application of Situated Design is to provide support for human information seeking behaviour in order to cope with the information load (Lueg, 1997; Lueg, 1999). From a situated perspective on human-machine interface, designing in the context of the Internet services, a situated information filtering through an adaptive

USENET interface was introduced. Also, there is ongoing research focusing on the concepts of situatedness of representations (Stahl, 1993; Wheeler, 1994; Rosenschein and Kaelbling, 1995; Morrison, 1998), situated interaction (Oehlmann et al., 1995; Rappaport, 1998), information retrieval (Hert, 1997) and situated autonomous agents (Maes, 1990; Errico and Aiello, 1996; Müller and Pecchiari, 1996; Beyer, 1998; Grecu and Brown, 1998; Gero and Fujii, 1999).

Chapter 3

Multiple Representations

A Platform for Situated Learning in Designing

"Realities have many different properties or attributes and each representation abstracts only a finite subset" (Akin, 1986)

This Chapter introduces the concept of multiple representations and its use in designing while designers construct their design artefacts. Different representations are needed in the design process when considering different issues. Some specific representations favour specific outcomes and since it is not known in advance which outcomes may be required so it is not known in advance which representation to use. The complexity of the situation is hard to grasp when only a single representation is available. This is particularly true when a rich representation of the situation is required and the represented situation is dynamic. The processes of developing multiple representations of a single object are introduced. An example of developing multiple representations within the domain of architectural shapes in the form of a floor plan is presented. The Chapter proposes the development of multiple representations as a platform for situated learning in designing.

The ability to adequately represent a design is crucial to reasoning about it. Representations encompass a wide range of possible ways to store information about an object, its function, behaviour and structure. Designing is concerned with the generation of design descriptions of potential artefacts in response to statements about the qualities to be exhibited by those artefacts (Coyne et al., 1990). A useful characterisation of designing is introduced within a framework model built around function, behaviour and structure (Gero, 1990). The work in this thesis focuses on representing knowledge of design artefacts within the domain of architectural shapes in the form of floor plans. This knowledge includes decompositional knowledge, physical properties and structural knowledge of shapes. Decompositional knowledge arises from the fact that an object may be composed of parts, each of which may be composed of subparts, and so on. A grouping of parts may be thought of as a component of an object. Physical properties are represented by describing an elementary object in terms of its geometrical properties. Components may have similar properties, which are derived from the properties of their parts. Structural knowledge concerns the nature of the interconnections or relationships between the components (Babin and Loganantharaj, 1991). In designing, these relationships assume a central role. For example, in

architectural plans, the individual components are usually not as interesting as the relationships among them (Mackellar and Peckham, 1992).

3.1 Multiple Representations while Designing

A design brief is an input into the design process. The design brief describes the client's requirements. The output of the design process is generally a set of contract documents, consisting of specifications and drawings. Thus, both the input and the output of the design process are representations that describe and depict an artefact or process. So, designing at some abstract level may be treated as the process of transforming one set of representations, the design brief, into another set of representations, the contract documents. However, not only the inputs and outputs of the design process are representations but the intervening transformations are also carried out on representations. Because designing typically occurs in situations where it is not possible or feasible to manipulate the environment directly, designers manipulate representations of the environment (Goel, 1995). Designers often employ standard transformations to manipulate structures. A transformation may be defined as an operation which changes one representation into another, while preserving certain properties. Architects are familiar with geometric transformation, rotation and reflection which may be employed to position instances of structures in compositions.

Representations may include objects and relationships, that establish significant links from one object to another; moreover, the representation used may influence the result obtained. Each representation is usually associated with a range of desired applications and is a partial view of the object it represents (Davis et al., 1993). This partial view is an interpretation of the object often aimed at a particular application or purpose. There is no one representation that allows detailed consideration of all possible concerns. One way to represent diversity is through the use of multiple representations. So it is often convenient and sometimes necessary to use a number of different representations. Equally important, in support of multiple representations, is that some specific representations favour specific outcomes and since it is not known in advance which outcomes may be required so it is not known in advance which representation to use. Using the approach of multiple representations is one way of overcoming this problem. In such an approach, each representation is suitable for a certain class of functions, because it allows a different interpretation of what has been drawn (Damski, 1996). Some of the roles that multiple representations play in designing include (Goldschmidt, 1997): interpretation, transformation and emphasis. The complexity of a situation is hard to grasp when only a single representation is available. This is particularly true if a rich representation of the situation is required.

The multiple representations approach is a cognitive strategy that designers and solvers of ill-structured problems adopt, because it facilitates the intricate process of creating links (Goldschmidt, 1997). According to Schön (1983), the designer is engaged in "graphical conversation with the design" and according to Lawson (1980), "the designer has a conversation with the drawing". These transactions with the external representation illuminate the visual mental process of designers. Where graphic media, such as design sketches, are used by the designer, design moves are "a series of actions"

of the designer which result in transformations of a representation (Lawson, 1980; Akin, 1986). In the process of conceptual designing, various moves by designers are encountered and expressed in graphical design development in the form of different representations reflecting transitions of states from one representation to the other. Typical processes can be interpreted as types of modification of the representation. For instance, the redescription process is one way in which designers cognitively exploit graphical representations (Oxman, 1995). In the re-representation process (a psychological theory of creativity), (Karmiloff-Smith, 1993), human beings explore new modifications through the externalisation of knowledge structures in representations. Re-representation of a form is possible only after an underlying representational structure of that form has been externalised and made explicit in the drawing. It also provides a changing pattern in the graphical medium. According to Arnheim (1969), perceptual cognitive operations in designing include distinguishing structural relationships of the images in the design representation. This includes the interpretation and conceptualisation of structural relationships. Spatial Gestalt and structural qualities are significant in conceptual designing.

Oxman (1997) demonstrated the significance of the concept of multiple representations in designing. According to Oxman (1997), the designer appears to be capable of exploiting these various underlying representational structures. Designers constructed explicit representational structures that were implicit in a particular design form (in this thesis, an architectural floor plan), during their moves as the design evolved. She reported her findings from an experiment aimed at studying the way designers deal with the richness of information in graphical representations. Designers were found to be able to extract various abstracted explicit representations of the existing design based on their own domain of knowledge. These abstractions include typological, organisational and morphological principles and functional relationships. They were able to externalise and categorise their graphic manipulations conceptually despite the fact that drawings tend not to provide explicit representational information to support such decomposition. Oxman (1997) concluded that designers utilise multiple representations of underlying conceptual structures. These multiple representations serve the designer as supportive representations for design manipulations. It is the structuring quality of these multiple representations which appear to support design manipulation.

Schön and Wiggins (1992), based on the analysis of a number of design protocols, suggest that sketching presents a visual display which can be potentially perceived in different ways; that is, the sketch can be reinterpreted. These perceptual re-interpretations produce a drawing episode which appears to indicate that the drawing is based on the image in the “mind’s eye” that resulted from the interpretations of the original image. Perceptual interpretations are referred to as "moves" while the judgements of the consequences and implications of the move are referred to as "seeing". They argue that designing consists of sequences of seeing-moving-seeing with the unintended consequences of moves allowing the designer to bring more facets of their knowledge into conscious thought allowing them to handle the complexity associated with ill-defined problems. They argue further that it is the interconnectedness between the various domains of knowledge relevant to a design that brings the unintended consequences of a move into consciousness and that it is the degree of interconnectedness which differentiates between an expert's and novice's design

knowledge. This conception is very similar to Goldschmidt's view (Goldschmidt, 1991; Goldschmidt, 1994).

Multiple representations can be viewed as multiple seeing through the concept of seeing-moving-seeing. These ways of seeing involve an appreciation process to see what needs to be focussed on from what is there at the time of seeing (Schön and Wiggins, 1992). For instance, top-down processing affects the way geometric features of shapes can be seen as in the case of Figure 3.1. For many people the triangles seem to point to the right. But if we try to perceive them as pointing upward and slightly left, this can be done with ease. Or, we can perceive the triangles as pointing downward and left. While doing so, the external representation by itself does not change but rather the representations we construct from it are changing depending upon our focus of what is there in the external representations.

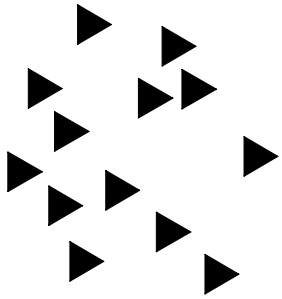


Figure 3.1 Look at the display of triangles, in which directions do they point? Can you perceive the direction differently? (Solso, 1997).

In the conceptual aspects of designing, multiple representation could play a critical role. It provides opportunities for designers to conceptualise their designs differently. Using multiple representations allows for different interpretations of what has been drawn. In designing, representations are constructed on the fly during the course of perception and no single representation is firmly and intentionally known in advance to be used. Different perceptions are akin to reformulations of the design artefact and consequently multiple representations are constructed. This allows seeing the design from different perspectives and offering different moves. Hence, different relationships between design elements across the representations may be discovered and learned. These relationships are beneficial in offering different ways of reasoning. Multiple representations and reasoning related to designing are important to endow the computer with the ability to learn these relationships during the design process. An example of how a design object (a square), can be seen and represented in many different ways is shown in Figure 3.2. A square can be represented as a set of four points; a set of four line 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 3.2(a) are shown in Figure 3.2(b). Each of which is suitable for one or more applications. The concept of multiple representations of a single object maps the fluid design environment. It also maps the "stepping out of representational flatland" in situated cognition (Clancey, 1991; Smith, 1999) in which representations are not stored, retrieved and manipulated but rather constructed in the course of perception where perception, cognition and learning are associated.

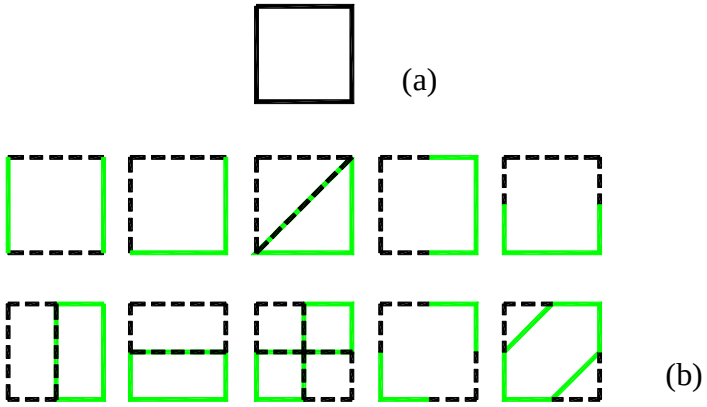


Figure 3.2 (a) Image of a square, (b) some of the possible representations of a square.

Landau (1996) suggested that objects could be represented in terms of different geometric descriptions or shape structures. Within the domain of architectural shapes, during the process of designing, a designer might encounter many different situations according to his focus of attention at each stage during the design process to reach the final product. Different representations of the design artefact may have been developed during the process of designing. Figure 3.3 illustrates the design descriptions of a final product: the plan, elevation and sections of Villa Capra, Italy. Different representations that could have been interpreted by the designer at different stages in the designing of Villa Capra are illustrated in Figure 3.4.

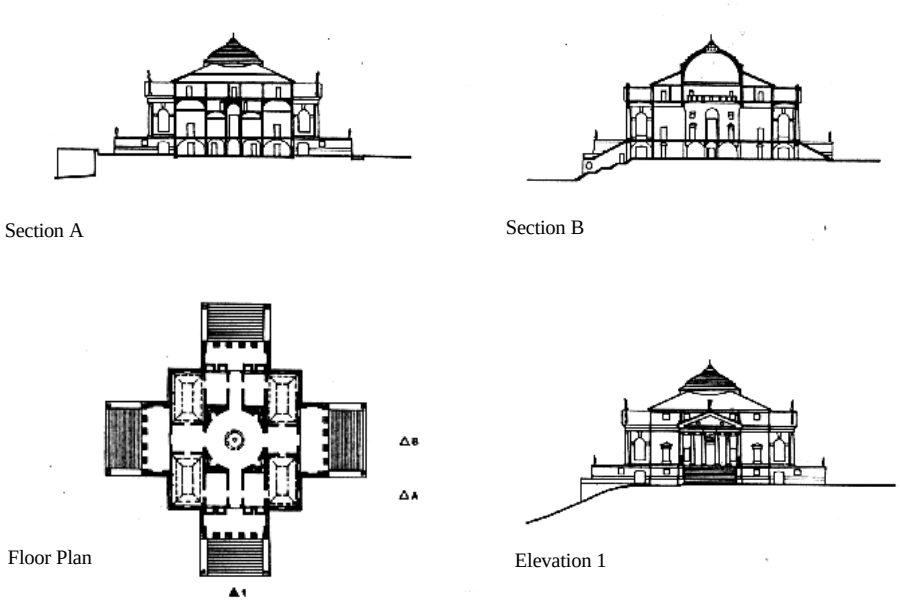


Figure 3.3 Various descriptions of Villa Capra, Vicenza, Italy (Clark and Pause, 1996).

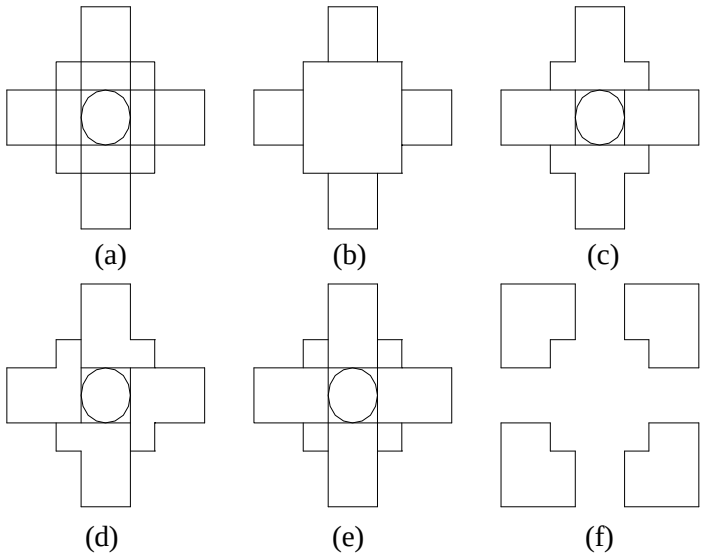


Figure 3.4 Some of the possible representations that might be interpreted by the using the platform proposed in this thesis: (a) lines, (b) blocks, (c) reflected components, (d) rotated components, (e) centrality and (f) background/foreground.

3.2 Multiple Representations of an Architectural Shape

Drawings have been used to represent physical and abstract ideas, simple and complex knowledge, local and universal information. The most consistent finding in the design area is that drawings are associated with reinterpretation or the emergence of new ways of seeing the drawing. The process of designing involves a recursive sequence of activities involving, thinking, imagery, drawing, reinterpretation and the access of different types of knowledge. The knowledge accessed can be perceptual and associated with the physical attributes of the design represented in a drawing (Purcell and Gero, 1998).

Fundamental to any computer-aided system in designing is the need to represent objects under consideration. Currently, CAD systems use numerical representations for drawings. This representation limits designers to use these systems only as drawing platforms that are not helpful at a more conceptual level of designing. Further, such representational systems do not readily lend themselves to some of the shape analysis tasks required by designers, such as topological relationships and shape transformations. Symbolic representation is dimensionless when compared to Cartesian spaces. When those objects are conceptual, a strictly symbolic representation is sufficient. It endows computer-aided systems in designing with basic tools for conceptual reasoning such as topological reasoning, directional reasoning, and shape emergence (Gero and Yan, 1994; Damski, 1996).

3.2.1 Initial representation of a shape

One of the forms of the designing outcome is a graphical description of a design artefact. In the basic case, this graphical description consists of line drawings that determine the shape. Stiny (1980) defines the shape with respect to a Cartesian coordinate system in which coordinates are used as primitives to represent the shape that is made up of line segments and a set of labelled points. In this way, a line segment is described by the coordinates of its two endpoints. This method of representing shapes as a set of discrete endpoints may not provide an appropriate foundation for shape recognition and does not cope with emergent subshapes because geometric properties of shape are not explicitly applied and are dependant upon calculating accuracy (Tan, 1990). The concept of construction lines has been used to represent shapes where shapes are represented as contiguous line segments connected at end-points or intersection as in a chain (Tan, 1990). As a result, emergent shapes are defined as a list of contiguous line segments.

In this thesis, infinite maximal lines are used as representational primitives to construct a symbolic representation as an initial representation of the shape to support shape recognition. Gero (1992b) and Gero and Yan (1993) have developed the notion of infinite maximal lines for representing a shape. This symbolic representation has been successfully implemented for discovering emergent shapes in two- and three-dimensional domains (Damski and Gero, 1994a; Gero and Yan, 1994). A line segment in a shape is maximal whenever no other line segment in the shape contains it. An extended maximal line is a line segment within which at least one maximal line is embedded. An infinite maximal line is the infinite line in which an extended maximal line is embedded. There are three kinds of properties of interest of infinite maximal lines: topological properties, geometrical properties and dimensional properties.

Drawings may be described in different ways by decomposing them into parts that may be ordered hierarchically or in some other way, and by assigning these parts to categories to clarify intention from different points of view (Stiny, 1990b). For demonstrating the development of a set of possible representations from the initial representation of a shape, the outline of the entries and the hexagon hall of the Sepulchral Church, Sir John Soane, 1796 (Jun, 1997) as shown in Figure 3.5 was chosen as an example. Using infinite maximal lines as representational primitives, the general form of the symbolic representation of shapes is:

$$\mathbf{S}_i = \{N_I ; [i_{jk}]\}$$

where N_I *is* the number of infinite maximal lines constituting a shape $\mathbf{S}_I$; and $[i_{jk}]$ is the description of the intersections of infinite maximal lines defining that shape. Alternatively, $\mathbf{S}_i = \{N_I ; [l_i]\}$ where $[l_i]$ is the description of the constraints on infinite maximal lines defining that shape: topological, geometrical and dimensional. Topological constraints concern the structure within which intersections and line segments are organised. Geometrical constraints on infinite maximal lines relate to their parallelarity, perpendicularity and skewness. Dimensional constraints are in terms of lengths of line segments and their proportions to each other. An example is presented in Figure 3.6.

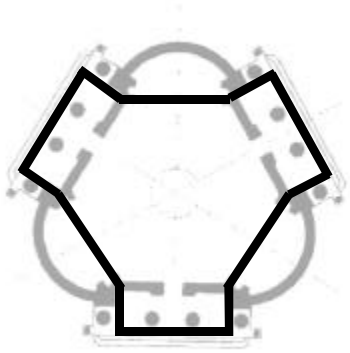
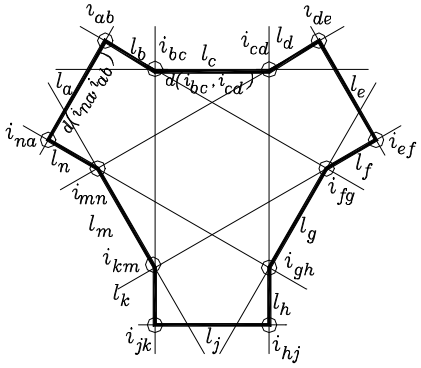


Figure 3.5 The outline of the entries and the hexagon hall of the Sepulchral Church, Sir John Soane, 1796, (Jun, 1997).



$\mathbf{S_i = \{N_l ; [i_{jk}]\}}$

$\mathbf{S_1}$ = {12; [i_{ab}, i_{bc}, i_{cd}, i_{de}, i_{ef}, i_{fg}, i_{gh}, i_{hj}, i_{jk}, i_{km}, i_{mn}, i_{na}]}

$\mathbf{S_i = \{N_l ; [l_i]\}}$

$\mathbf{S_1}$ = {12; [l_a, l_b, l_c, l_d, l_e, l_f, l_g, l_h, l_j, l_k, l_m, l_n],

l_a // l_g, l_b // l_n, l_c // l_j, l_d // l_f, l_e // l_m, l_h // l_k,

$l_n \perp l_a$, $l_a \perp l_b$, $l_d \perp l_e$, $l_e \perp l_f$, $l_h \perp l_j$, $l_j \perp l_k$,

$A(l_m, l_n) = A(l_b, l_c) = A(l_c, l_d) = A(l_f, l_g) = A(l_g, l_h) = A(l_k, l_m)$,

$d(i_{na}, i_{ab}) = d(i_{de}, i_{ef}) = d(i_{hj}, i_{jk})$, $d(i_{ab}, i_{bc}) = d(i_{cd}, i_{de}) = d(i_{ef}, i_{fg}) =$

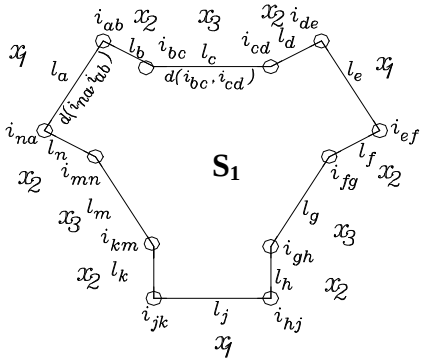
$d(i_{gh}, i_{hjj}) = d(i_{jk}, i_{km}) = d(i_{mn}, i_{na})$, $d(i_{bc}, i_{cd}) = d(i_{fg}, i_{gh}) = d(i_{km}, i_{mn})$}

Where	A	= angle between two line segments
	l_i	= infinite maximal lines
	d	= distance or length between two intersections of infinite maximal lines
	i_{jk}	= intersection of two infinite maximal lines l_j and l_k
	//	= lines are parallel
	$\perp$	= lines are perpendicular

Figure 3.6 Symbolic representation using infinite maximal line, l_a to l_n, as representational primitives.

The similarity among line segments is one way of grouping the structural elements of line segments in the initial representation. The similarity measurements are based upon

the distances between the intersections of each two maximal lines defining line segments on the boundary of the shape, distances from the centre of that line segment and the centre of the whole shape. Labelling of line segments based on this kind of similarity is illustrated in Figure 3.7.



$\mathbf{S_1}$ = {12; [l_a , l_b , l_c , l_d , l_e , l_f , l_g , l_h , l_j , l_k , l_m , l_n]}

The lengths of line segments embedded in l_a, l_e and l_j are equal and their distances from the centre of the shape are also equal; and we have labelled them as x_1. The same with both l_b, l_d, l_f , l_h, l_k and l_n which we have labelled as x_2 and l_c , l_g and l_m labelled as x_3. The following notations are used in the syntax of representations: $\boldsymbol{N_i}$ **::** $\mathbf{S_j}$ is the re-representation number *i* for the initial representation of the shape $\mathbf{S_j}$, "( )" indicates a group of line segments that are unbounded and "[ ]" indicates a group of line segments that are bounded which means the line segments are sequentially connected to form a closed shape.

$\mathbf{S_1}$= {12; [x_1, x_2, x_3, x_2, x_1, x_2, x_3, x_2, x_1, x_2, x_3, x_2]}

Figure 3.7 Labelling line segments with x_1, x_2 and x_3 based on similarity measurements.

3.2.2 Development of multiple representations

Alternate representation makes new interpretations possible. Interpretations are only implicit in an existing representation (Damski and Gero, 1994b). Interpretation is the process of inferring results from a given object in a particular representation. Therefore, re-representation allows implicit properties in one representation to become explicit in other representations. Different representations may be derived from the same initial representation and can serve as a foundation upon which to develop many different kinds of inference. Drawings such as architectural floor plans can be considered informal representations, based on a loose set of conventions, in which multiple representations are embedded and can be derived (Chase, 1993).

There are different ways of developing multiple representations of an initial representation of a shape. The processes utilised here for developing multiple representations, as shown in Figure 3.8, commence with representing a design composition in an initial representation as shown in Figure 3.6. This initial representation of the shape may be treated as being composed of many sub-elements. The composition of sub-elements can be interpreted differently. Each individual representation could be composed of repeated or common structural properties in a

group of objects where a single object in a group is represented in terms of different attributes. Developing multiple representations can be through decomposing the initial representation to its components and grouping the objects based upon the commonality of their structural properties. The decomposition of design drawing into different structures of knowledge is a decomposition of different types of knowledge. Design drawings are not readily decomposable into separate representations, or structures. In order to isolate implicit representational structures it is important to identify the explicit representations which support visual reasoning. Since drawings are not explicitly structured according to these representations, it is significant to externalise them in re-representation. The decomposition in the initial representation can be of either parts of shape boundary or shape areas. The decomposition of the shape boundary into parts results in line segments that are regrouped alternatively based on the commonality of their structural properties. This leads to developing a set of representations of unbounded *n*-sided shapes, as shown in Section 3.2.2.1.

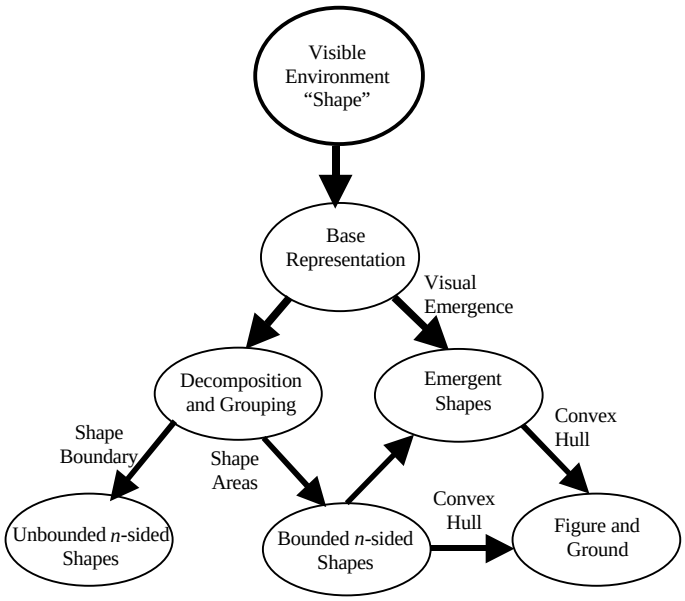


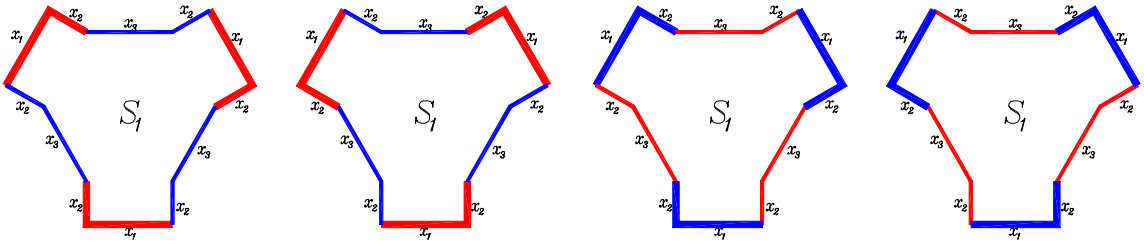
Figure 3.8 Different processes of developing multiple representations of a single shape.

Other ways of developing multiple representations from the initial representation include changing the representation of that shape through structure interpretation. Structural interpretation concerns changing the original shapes into new ones by modifying their structures. Structure is concerned with the components of objects and their relationships (Gero, 1990). In this process new structural properties that were implicit in the initial representation may become explicit in the changed representations. This has been defined as one form of visual emergence. The decomposition of the boundary of the design composition into areas within that boundary results in sets of groups of bounded *n*-sided shapes where emergent shapes could appear. This may be achieved through alternative connections between the vertices of the shape. A set of representations can be developed from grouping these bounded *n*-sided shapes based upon their congruency as shown in Section 3.2.2.2.

Alternatively visual emergence could be facilitated to develop a set of multiple representations. One way for visual emergence to occur is through grouping the shapes produced through the intersections of line segments of the infinite maximal lines of the shape as shown in Section 3.2.2.3. Furthermore, the concept of figure and ground is used to develop alternative representations as shown in Section 3.2.2.4. Using these different ways, we have developed 28 representations of the exemplar shape as a set of some of the possible representations from the initial representation of the shape.

3.2.2.1 Unbounded *n*-sided subshapes representations

Unbounded *n*-sided subshapes are created through decomposing the boundary of the shape into line segments and regrouping line segments that are connected to each other forming open shapes based upon the commonality of their structural properties. In unbounded two-sided subshapes the similarity measurements are based upon the repetitions of two contiguous line segments, distances between the centres of their maximal lines to the centre of the whole shape. Thus, other ways to re-represent the shape from its initial representation are shown in Figures 3.9(a) to 3.9(d).

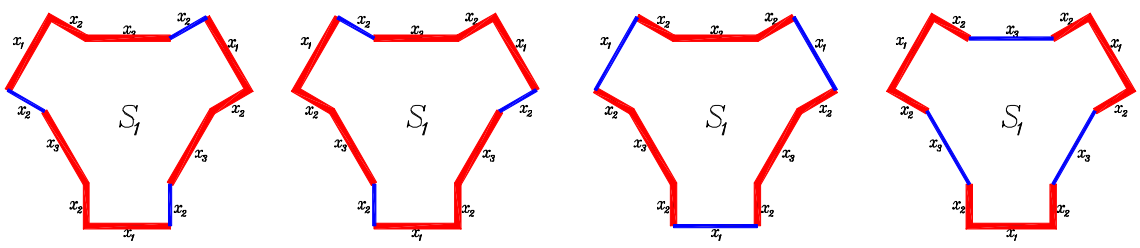


(a): N_1 (b): N_2 (c): N_1 (d): N_2

$\mathbf{S_1}$ = {[$x_1, x_2, x_3, x_2, x_1, x_2, x_3, x_2, x_1, x_2, x_3 , x_2$]}
N_1 :: $\mathbf{S_1}$ = {[3((x_1, x_2), (x_3, x_2))]}
N_2 :: $\mathbf{S_1}$ = {[3((x_2, x_3), (x_2, x_1))]}

Figure 3.9 Two representations, N_1 and N_2, of unbounded two-sided subshapes; (a) and (c) have the same description of the representation N_1; (b) and (d) have the same description of the representation N_2.

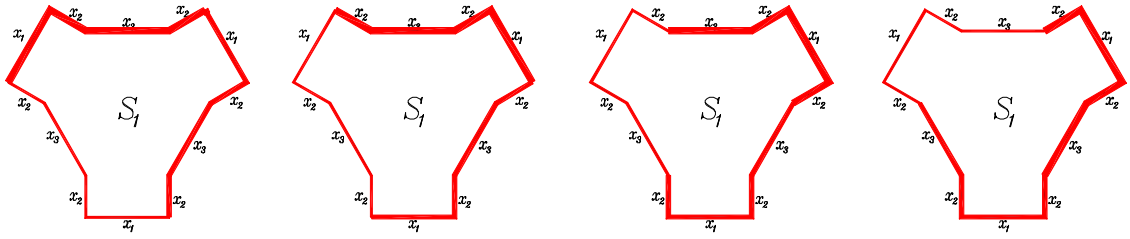
The same similarity measures could be applied with three contiguous line segments. Examples of different representations of unbounded three-sided subshapes are presented in Figures 3.10(a) to 3.10(c). Similarly, other representations of four and five-sided unbounded shapes are as shown in Figures 3.11 and 3.12 respectively.



(a): N_3 (b): N_4 (c): N_5 (d): N_6

$\mathbf{S_1}$ = {[x_1, x_2, x_3, x_2, x_1, x_2, x_3, x_2, x_1, x_2, x_3, x_2]}
$\boldsymbol{N_4}$ :: $\mathbf{S_1}$ = {[3((x_1, x_2, x_3), (x_2))]}
$\boldsymbol{N_5}$ :: $\mathbf{S_1}$ = {[3((x_3, x_2, x_1), (x_2))]}
$\boldsymbol{N_6}$ :: $\mathbf{S_1}$ = {[3((x_1), (x_2, x_3, x_2))]}
$\boldsymbol{N_7}$ :: $\mathbf{S_1}$ = {[3((x_2, x_1, x_2), (x_3))]}

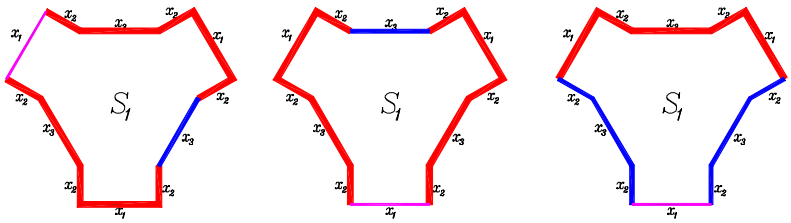
Figure 3.10 Four representations, $\boldsymbol{N_3}$ to $\boldsymbol{N_6}$, of bounded three-sided sub-shapes.



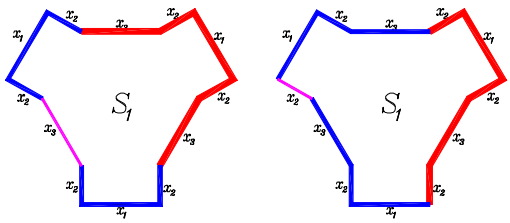
(a): $\boldsymbol{N_7}$ (b): $\boldsymbol{N_8}$ (c): $\boldsymbol{N_9}$ (d): $\boldsymbol{N_{10}}$

$\mathbf{S_1}$= {[x_1, x_2, x_3, x_2 , x_1, x_2, x_3, x_2 , x_1, x_2, x_3 , x_2]}
$\boldsymbol{N_7}$ :: $\mathbf{S_1}$ = {[3(x_1, x_2, x_3, x_2)]}
$\boldsymbol{N_8}$ :: $\mathbf{S_1}$ = {[3(x_2, x_3, x_2, x_1)]}
$\boldsymbol{N_9}$ :: $\mathbf{S_1}$ = {[3(x_3, x_2, x_1, x_2)]}
$\boldsymbol{N_{10}}$ :: $\mathbf{S_1}$ = {[3(x_2, x_1, x_2, x_3)]}

Figure 3.11 Four representations, $\boldsymbol{N_7}$ to $\boldsymbol{N_{10}}$, of unbounded four-sided subshapes.



(a): $\boldsymbol{N_{11}}$ (b): $\boldsymbol{N_{12}}$ (c): $\boldsymbol{N_{13}}$



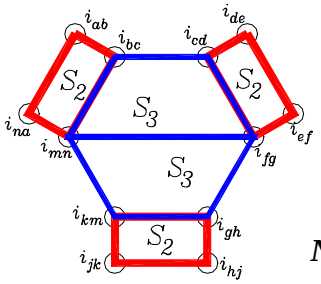
(d): $\boldsymbol{N_{14}}$ (e): $\boldsymbol{N_{15}}$

$\boldsymbol{N_{11}}$ :: $\mathbf{S_1}$ = {[x_1, (x_2, x_3, x_2, x_1, x_2), x_3, (x_2, x_1, x_2, x_3 , x_2)]}
$\boldsymbol{N_{12}}$ :: $\mathbf{S_1}$ = {[x_1, x_2), x_3, (x_2, x_1, x_2, x_3, x_2), x_1, (x_2, x_3, x_2]}
$\boldsymbol{N_{13}}$ :: $\mathbf{S_1}$ = {[(x_1, x_2, x_3, x_2, x_1), (x_2, x_3 , x_2), x_1, (x_2 , x_3 , x_2)]}
$\boldsymbol{N_{14}}$ :: $\mathbf{S_1}$ = {[x_1, x_2), (x_3, x_2, x_1, x_2, x_3), (x_2, x_1 , x_2), x_3 , (x_2]}
$\boldsymbol{N_{15}}$ :: $\mathbf{S_1}$ = {[(x_1, x_2, x_3), (x_2, x_1, x_2, x_3, x_2), (x_1, x_2, x_3), x_2]}

Figure 3.12 Five representations, $\boldsymbol{N_{11}}$ to $\boldsymbol{N_{15}}$, of unbounded five-sided subshapes.

3.2.2.2 Bounded *n*-sided subshapes representations

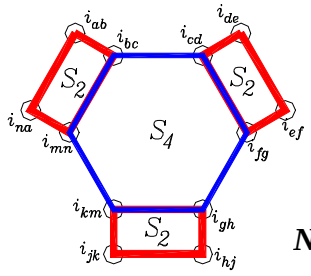
Bounded *n*-sided subshapes are generated through decomposing the initial representation of the shape into areas or subshapes within the boundary of the shape. The decomposition can be through connecting the vertices of the shape alternatively to create circuits (*n*-sided bounded subshapes). The concept of re-representing the shape is different here because the structural knowledge in the initial representation is changed. Some structural elements embedded in the initial representation were only implicit. As a result new subshapes that existed only implicitly in the shape before and have never been explicitly indicated are emerged. Emergence is defined as the process of making explicit unexpected properties that previously were only implicit (Gero et al., 1995). Examples of bounded four-sided subshapes (emergent shapes), consisting of various numbers of line segments joined together to form closed shapes are [x_2, x_1, x_2, x_1] labelled as [S_2] and [x_3, x_1, x_4, x_1] labelled as [S_3] as shown in Figures 3.13. In this representation and onwards notice that different labels of other line segments such as x_4 have been introduced. This is because the length embedded in its line segments is not equal to any of those used before and the distance from its centre to the centre of the whole shape is different. The subshapes that were created alternatively are grouped based on their congruency. A representation consists of a mixture of bounded four-sided shapes and the remaining parts from the original shape is shown in Figure 3.14. Different examples of representations from mixtures of three and four-sided shapes are shown in Figure 3.15.



***N*16**

S1 = {12; [i_{ab}, i_{bc}, i_{cd}, i_{de}, i_{ef} , i_{fg}, i_{gh}, i_{hj}, i_{jk}, i_{km} , i_{mn}, i_{na}]}
S1 = {[i_{mn}, i_{na}, i_{ab}, i_{bc}], [i_{cd}, i_{de}, i_{ef}, i_{fg}], [i_{gh}, i_{hj}, i_{jk}, i_{km}], [i_{bc}, i_{cd}, i_{fg}, i_{mn}], [i_{gh}, i_{km}, i_{mn}, i_{fg}]}
S1 = {[x_2, x_1, x_2, x_1], [x_2, x_1, x_2, x_1], [x_2, x_1, x_2, x_1], [x_3, x_1, x_4, x_1], [x_3, x_1, x_3, x_4]}
S1 = {[S_2], [S_2], [S_2], [S_3], [S_3]}
***N*16** **:: S1** = {3[S_2], 2[S_3]}

Figure 3.13 An example of a representation ***N*16** consisting of bounded four-sided subshapes.

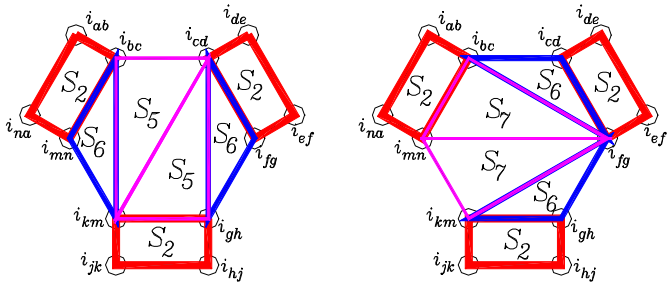


***N*17**

S1 = {[i_{mn},i_{na},i_{ab}, i_{bc}], [i_{cd} , i_{de} , i_{ef} , i_{fg}], [i_{gh}, i_{hj} , i_{jk} , i_{km}], [i_{bc}, i_{cd}, i_{fg}, i_{gh}, i_{km} , i_{mn}]}
S1 = {[x_2, x_1, x_2, x_1], [x_2, x_1, x_2, x_1], [x_2, x_1, x_2, x_1], [x_3, x_1, x_3, x_1,x_3, x_1]}

$\mathbf{S_1}$ = {[S_2], [S_2], [S_2], [S_4]}
$\boldsymbol{N_{17}}$ :: $\mathbf{S_1}$ = {3[S_2], [S_4]}

Figure 3.14 An example of a representation $\boldsymbol{N_{17}}$ consisting of bounded four-sided subshapes and the remaining part of the initial shape.



(a): $\boldsymbol{N_{18}}$ (b): $\boldsymbol{N_{19}}$

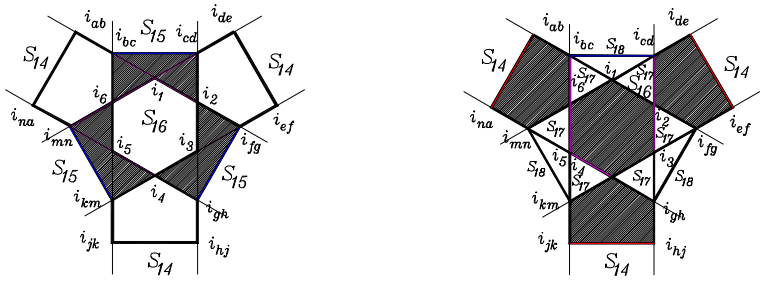
$\mathbf{S_1}$ = { [i_{mn},i_{na},i_{ab}, i_{bc}],[i_{cd} , i_{de}, i_{ef} , i_{fg}], [i_{gh}, i_{hj} , i_{jk} , i_{km}], [i_{mn}, i_{bc}, i_{km}], [i_{cd}, i_{fg}, i_{gh}], [i_{bc}, i_{cd}, i_{km}], [i_{cd}, i_{gh}, i_{km}]}
$\mathbf{S_1}$ = { [x_2, x_1, x_2, x_1], [x_2, x_1, x_2, x_1], [x_2, x_1, x_2, x_1], [x_1, x_5, x_3], [x_1, x_3, x_5], [x_1, x_4, x_5], [x_5, x_1, x_4]}
$\mathbf{S_1}$ = { [S_2], [S_2], [S_2], [S_5], [S_5], [S_6], [S_6]}
$\boldsymbol{N_{18}}$ :: $\mathbf{S_1}$ = {3[S_2], 2[S_5], 2[S_6]}

$\mathbf{S_1}$= { [S_2], [S_2], [S_2], [S_6], [S_6], [S_7], [S_7]}
$\boldsymbol{N_{19}}$ :: $\mathbf{S_1}$ = {3[S_2], 2[S_6], 2[S_7]}

Figure 3.15 Two representations $\boldsymbol{N_{18}}$ and $\boldsymbol{N_{19}}$ are mixtures of bounded three and four-sided subshapes.

3.2.2.3 Emergent shapes

Visual emergence is the process of "seeing" visual structures that are not explicitly represented. As a consequence, other representations can be developed (Gero et al., 1995; Jun and Gero, 1998). Emergent shapes are introduced through interpretative and perceptual processes concerned with arriving at alternative descriptions of the shape. A transformational process uses the existing pattern for generating new structures in a variety of ways (Soufi and Edmonds, 1996). Alternative groupings of subshapes consisting of the intersections of infinite maximal lines provide ways of arriving at alternative descriptions of the shape and generate new structures under which new emergent shapes are developed. Examples of emergent shapes are illustrated in Figures 3.16(a) to 3.16(g) that show emergent shapes as a result of a process of visual emergence but not necessarily as may be perceived by human observers.



(a): $\boldsymbol{N_{20}}$ (b): $\boldsymbol{N_{21}}$

$\mathbf{S_1}$ = {[i_{mn}, i_{na}, i_{ab}, i_{bc}, i_{cd}, i_{de}, i_{ef}, i_{fg}, i_{gh}, i_{hj}, i_{jk}, i_{km}]}

$\mathbf{S_1}$ = {[i_{na}, i_{ab}, i_{bc}, i_6, i_{mn}], [i_{bc}, i_{cd}, i_2, i_1, i_6], [i_{de}, i_{ef}, i_{fg}, i_2 , i_{cd}], [i_{fg}, i_{gh}, i_4, i_3, i_1], [i_{hj}, i_{jk}, i_{km}, i_4, i_{gh}], [i_{km}, i_{mn}, i_6, i_5, i_4], [i_1, i_2, i_3, i_4 , i_5, i_6]}

$\mathbf{S_1}$ = {[x_1, x_2, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}, x_{11}, x_{11}], [x_1, x_2, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}, x_{11}, x_{11}], [x_1, x_2, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}, x_{11}, x_{11}], [x_{11}, x_{11}, x_{11}, x_{11}, x_{11}, x_{11}]}

$\mathbf{S_1}$ = {[S_{14}], [S_{15}], [S_{14}], [S_{15}], [S_{14}], [S_{15}], [S_{16}]}

$\boldsymbol{N_{20}}$ **::** $\mathbf{S_1}$ = {3[S_{14}], 3[S_{15}], [S_{16}]}

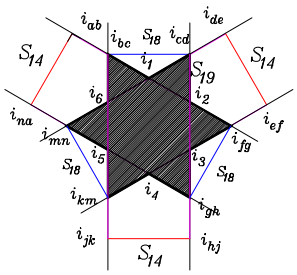
$\mathbf{S_1}$ = {[i_{mn}, i_{na}, i_{ab}, i_{bc} , i_{cd} , i_{de} , i_{ef} , i_{fg}, i_{gh}, i_{hj}, i_{jk}, i_{km}]}

$\mathbf{S_1}$ = {[i_{na}, i_{ab}, i_{bc}, i_6 , i_{mn}], [i_{bc}, i_{cd}, i_1], [i_{cd}, i_2, i_1], [i_{bc}, i_1, i_4], [i_{de}, i_{ef}, i_{fg}, i_2, i_{cd}], [i_{fg}, i_{gh}, i_3], [i_{fg}, i_3, i_2], [i_{gh}, i_3, i_6], [i_{hj}, i_{jk}, i_{km}, i_4 , i_{gh}], [i_{km}, i_{mn}, i_5], [i_{km}, i_4, i_5], [i_{mn}, i_5, i_6], [i_1, i_2, i_3, i_4 , i_5 , i_6]}

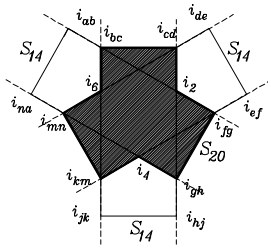
$\mathbf{S_1}$ = {[x_1, x_2, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}], [x_{11}, x_{11}, x_{11}], [x_{11}, x_{11}, x_{11}], [x_1, x_2, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}], [x_{11}, x_{11}, x_{11}], [x_{11}, x_{11}, x_{11}], [x_{11}, x_{11}, x_{11}, x_{11}, x_{11}, x_{11}]}

$\mathbf{S_1}$ = {[S_{14}], [S_{18}], [S_{17}], [S_{17}], [S_{14}], [S_{18}], [S_{17}], [S_{17}], [S_{14}], [S_{18}], [S_{17}], [S_{17}], [S_{16}]}

$\boldsymbol{N_{21}}$**::** $\mathbf{S_1}$ = {3[S_{14}], 3[S_{18}], 6[S_{17}], [S_{16}]}



(c): $\boldsymbol{N_{22}}$



(d): $\boldsymbol{N_{23}}$

$\mathbf{S_1}$ = {[i_{mn}, i_{na}, i_{ab}, i_{bc} ,i_{cd} , i_{de} , i_{ef} , i_{fg} ,i_{gh}, i_{hj} , i_{jk} , i_{km}]}

$\mathbf{S_1}$ = {[i_{na}, i_{ab}, i_{bc}, i_6 , i_{mn}], [i_{bc}, i_{cd}, i_1], [i_{de}, i_{ef}, i_{fg}, i_2 , i_{cd}], [i_{fg}, i_{gh}, i_3], [i_{hj}, i_{jk}, i_{km}, i_4 , i_{gh}], [i_{km}, i_{mn}, i_5], [i_{bc}, i_1, i_{cd}, i_2 , i_{fg} , i_3, i_{gh} , i_4 , i_{km}, i_5 , i_{mn} , i_6]}

$\mathbf{S_1}$ = {[x_1, x_2, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}], [x_1, x_2, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}], [x_1, x_2, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}], [x_{11}, x_{11}, x_{11}, x_{11}, x_{11}, x_{11} , x_{11}, x_{11}, x_{11}, x_{11}, x_{11}, x_{11}]}

$\mathbf{S_1}$ = {[S_{14}], [S_{18}], [S_{14}], [S_{18}], [S_{14}], [S_{18}], [S_{19}]}

$\boldsymbol{N_{22}}$ **::** $\mathbf{S_1}$ = {3[S_{14}], 3[S_{18}], [S_{19}]}

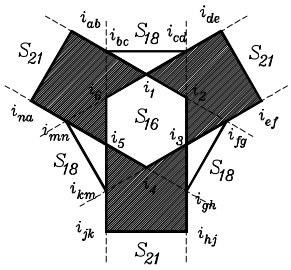
$\mathbf{S_1}$ = {[i_{mn},i_{na},i_{ab}, i_{bc} ,i_{cd} , i_{de} , i_{ef} , i_{fg} ,i_{gh}, i_{hj} , i_{jk} , i_{km}]}

$\mathbf{S_1}$ = {[i_{na}, i_{ab}, i_{bc}, i_6 , i_{mn}], [i_{de}, i_{ef}, i_{fg}, i_2 , i_{cd}], [i_{hj}, i_{jk}, i_{km}, i_4 , i_{gh}], [i_{bc}, i_{cd}, i_2 , i_{fg} , i_{gh} , i_4 , i_{km}, i_{mn} , i_6]}

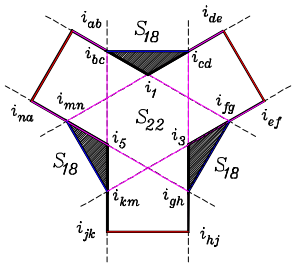
$\mathbf{S_1}$ = {[x_1, x_2, x_{11}, x_{11}, x_2], [x_1, x_2, x_{11}, x_{11}, x_2], [x_1, x_2, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}, x_3, x_{11}, x_{11} , x_3, x_{11}, x_{11}]}

$\mathbf{S_1}$ = {[S_{14}], [S_{14}], [S_{14}], [S_{20}]}

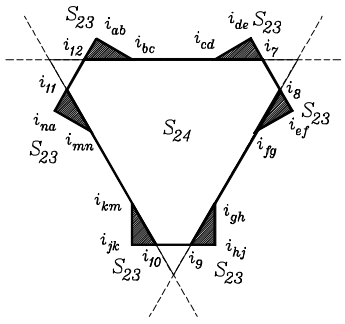
$\boldsymbol{N_{23}}$ **::** $\mathbf{S_1}$ = {3[S_{14}], [S_{20}]}



(e): $\boldsymbol{N_{24}}$



(f): $\boldsymbol{N_{25}}$



(g): $\boldsymbol{N_{26}}$

$\mathbf{S_1}$ = {[i_{mn},i_{na},i_{ab}, i_{bc} ,i_{cd} , i_{de} , i_{ef} , i_{fg} ,i_{gh}, i_{hj} , i_{jk} , i_{km}] }
$\mathbf{S_1}$ = {[i_{na}, i_{ab}, i_{bc}, i_I , i_6 , i_5 , i_{mn}], [i_{bc}, i_{cd}, i_1], [i_{de}, i_{ef}, i_{fg}, i_3 , i_2 , i_I , i_{cd}], [i_{fg}, i_{gh}, i_3], [i_{hj}, i_{jk}, i_{km}, i_4 , i_{gh}], [i_{km}, i_{mn}, i_5], [i_1, i_2, i_3, i_4 , i_5 , i_6]}
$\mathbf{S_1}$ = {[x_1, x_2, x_{11}, x_{11}, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}], [x_1, x_2, x_{11}, x_{11}, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}], [x_1, x_2, x_{11}, x_{11}, x_{11}, x_{11}, x_2], [x_3, x_{11}, x_{11}], [x_{11}, x_{11}, x_{11}, x_{11}, x_{11}, x_{11}]}
$\mathbf{S_1}$ = {[S_{21}], [S_{18}], [S_{21}], [S_{18}], [S_{21}], [S_{18}], [S_{16}]}
$\boldsymbol{N_{24}}$ **::** $\mathbf{S_1}$ = {3[S_{21}], 3[S_{18}], [S_{16}]}

$\mathbf{S_1}$ = {[i_{mn},i_{na},i_{ab}, i_{bc}, i_{cd} , i_{de} , i_{ef} , i_{fg}, i_{gh}, i_{hj} , i_{jk} , i_{km}] }
$\mathbf{S_1}$ = {[i_{mn}, i_{na}, i_{ab}, i_{bc}, i_I , i_{cd} , i_{de} , i_{ef} , i_{fg}, i_3, i_{gh}, i_{hj} , i_{jk} , i_{km}, i_5], [i_{bc}, i_{cd}, i_1], [i_{fg}, i_{gh}, i_3], [i_{km}, i_{mn}, i_5]}
$\mathbf{S_1}$ = {[x_2, x_1, x_2, x_{11}, x_{11}, x_2, x_1, x_2, x_{11}, x_{11}, x_2, x_1, x_2, x_{11}, x_{11}], [x_3, x_{11}, x_{11}], [x_3, x_{11}, x_{11}], [x_3, x_{11}, x_{11}]}
$\mathbf{S_1}$ = {[S_{22}], [S_{18}], [S_{18}], [S_{18}]}
$\boldsymbol{N_{25}}$ **::** $\mathbf{S_1}$ = {[S_{22}], 3[S_{18}]}

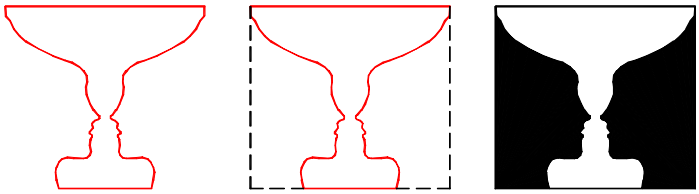
$\mathbf{S_1}$ = {[i_{mn},i_{na},i_{ab}, i_{bc}, i_{cd} , i_{de} , i_{ef} , i_{fg}, i_{gh}, i_{hj} , i_{jk} , i_{km}] }
$\mathbf{S_1}$ = {[i_{na}, i_{11}, i_{mn}], [i_{12}, i_{ab} , i_{bc}], [i_{cd} , i_{de}, i_7], [i_8, i_{ef} , i_{fg}], [i_{gh}, i_{hj}, i_9], [i_{10}, i_{jk}, i_{km}], [i_{mn}, i_{11}, i_{12}, i_{bc}, i_{cd}, i_7, i_8, i_{fg}, i_{gh}, i_9, i_{10}, i_{km}]}
$\mathbf{S_1}$ = {[x_{13}, x_2, x_{12}], [x_{13}, x_2, x_{12}], [x_{13}, x_2, x_{12}], [x_{13}, x_2, x_{12}], [x_{13}, x_2, x_{12}], [x_{13}, x_2, x_{12}], [x_{14}, x_{12}, x_3, x_{12}, x_{14}, x_{12}, x_3, x_{12}, x_{14}, x_{12}, x_3, x_{12}]}
$\mathbf{S_1}$ = {[S_{23}], [S_{23}], [S_{23}], [S_{23}], [S_{23}], [S_{23}], [S_{24}]}
$\boldsymbol{N_{26}}$ **::** $\mathbf{S_1}$ = {6[S_{23}], [S_{24}]}

Figure 3.16 Seven representations from $\boldsymbol{N_{20}}$ to $\boldsymbol{N_{26}}$ include emergent shapes as a result of an emergence process.

3.3.2.4 Figure and ground

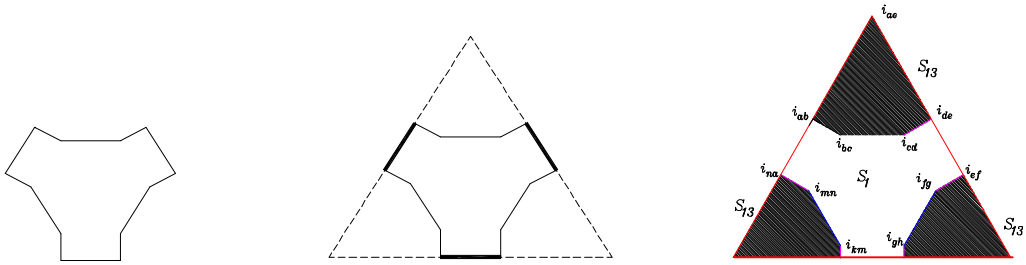
The figure and ground perception hypothesis developed in Gestalt psychology explores other aspects of how our visual system functions. It seems that our visual system simplifies the visual scene into a figure that we look at and a ground that is everything else in the scene that forms the background. For instance, the shape shown in Figure 3.17(a) could be perceived as a vase. The construction of the new representation of that shape could be through creating either a standard convex hull or an outermost convex hull of the initial shape. After constructing the representations shown in Figures 3.17(b) and 3.17(c) the shape could be perceived as a white central vase, or as a pair of black faces in profile that are looking towards each other. Generally when we see one of the perceptions, the other region forms a background and is not seen, so to see both precepts requires switching back and forth. The contour dividing the black and white regions of the picture appears to belong to whichever region is perceived as the figure. Figure and ground is a way of representing the illusory shapes as might be perceived by humans.

Two ways to develop representations under which the figure and ground perception can be introduced are through the standard convex hull or through the outermost convex hull of the initial shape. The outermost convex hull can be constructed by joining the farthest intersections of infinite maximal lines that represent the shape as shown in Figures 3.18(b) and 3.18(c). The standard convex hull can be constructed simply by joining the shape edges at the outline contour as shown in Figures 3.19(b) and 3.19(c).



(a) (b) (c)

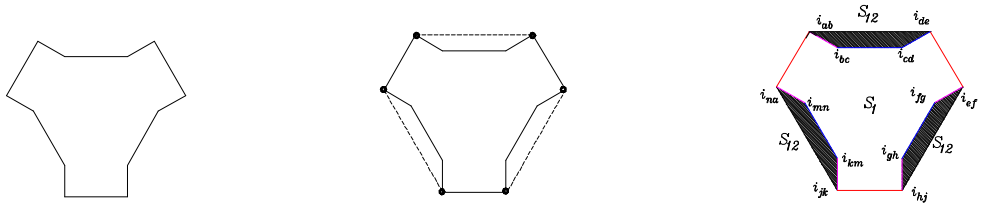
Figure 3.17 Figure and ground perception after (Bruce et al., 1996).



(a) (b) (c): $\boldsymbol{N_{27}}$

S = {[i_{mn}, i_{na}, i_{ab}, i_{bc}, i_{cd}, i_{de}, i_{ef}, i_{fg}, i_{gh}, i_{hj}, i_{jk}, i_{km}], [i_{na}, i_{bc}, i_{cd}, i_{de}, i_{ae}], [i_{ef}, i_{fg}, i_{gh}, i_{hj}, i_{ej}], [i_{jk}, i_{km}, i_{mn}, i_{na}, i_{ja}]}
S = {[x_1, x_2, x_3, x_2, x_1, x_2, x_3, x_2, x_1, x_2, x_3, x_2], [x_2, x_3, x_2, x_{10}, x_{10}], [x_2, x_3, x_2 , x_{10}, x_{10}], [x_2, x_3, x_2 , x_{10}, x_{10}]}
S = {[S_1], [S_{13}], [S_{13}], [S_{13}]}
$\boldsymbol{N_{27}}$ **:: S** = {[S_1], 3[S_{13}]}

Figure 3.18 Figure and ground representation using the outermost convex hull method



(a) (b) (c): $\boldsymbol{N_{28}}$

S = {[i_{mn}, i_{na}, i_{ab}, i_{bc}, i_{cd}, i_{de}, i_{ef}, i_{fg}, i_{gh}, i_{hj}, i_{jk}, i_{km}], [i_{na}, i_{bc}, i_{cd}, i_{de}], [i_{ef}, i_{fg}, i_{gh}, i_{hj}], [i_{jk}, i_{km}, i_{mn}, i_{na}]}
S = {[x_1, x_2, x_3, x_2, x_1, x_2, x_3, x_2, x_1, x_2, x_3, x_2], [x_2, x_3, x_2 , x_9], [x_2, x_3, x_2 , x_9], [x_2, x_3, x_2 , x_9]}
S = {[S_1], [S_{12}], [S_{12}], [S_{12}]}
$\boldsymbol{N_{29}}$ **:: S**$_1$ ={[S_1], 3[S_{12}]}

Figure 3.19 Figure and ground representation using the standard convex hull method.

The set of representations presented graphically in this Section is illustrated in a concise syntax as shown in Table 3.1.

Table 3.1 Concise syntax of a set of some possible representations from the initial representation shown in Figure 3.7.

No.	Multiple Representations	
	Label	**Concise syntax**
N_1	Unbounded two-sided subshapes	$\mathbf{S_1}$ = {[3((x_1, x_2), (x_3, x_2))]}
N_2		$\mathbf{S_1}$ = {[3((x_2, x_3), (x_2, x_1))]}
N_3	Unbounded three-sided subshapes	$\mathbf{S_1}$ = {[3((x_1, x_2, x_3), (x_2))]}
N_4		$\mathbf{S_1}$ = {[3((x_3, x_2, x_1), (x_2))]}
N_5		$\mathbf{S_1}$ = {[3((x_1), (x_2, x_3, x_2))]}
N_6		$\mathbf{S_1}$ = {[3((x_2, x_1, x_2), (x_3))]}
N_7	Unbounded four-sided subshapes	$\mathbf{S_1}$ = {[3(x_1, x_2, x_3, x_2)]}
N_8		$\mathbf{S_1}$ = {[3(x_2, x_3, x_2, x_1)]}
N_9		$\mathbf{S_1}$ = {[3(x_3, x_2, x_1, x_2)]}
N_{10}		$\mathbf{S_1}$ = {[3(x_2, x_1, x_2, x_3)]}
N_{11}	Unbounded five-sided subshapes	$\mathbf{S_1}$ = {[x_1, (x_2, x_3, x_2, x_1, x_2), x_3, (x_2, x_1, x_2, x_3 , x_2)]}
N_{12}		$\mathbf{S_1}$ = {[x_1, x_2), x_3, (x_2, x_1, x_2, x_3, x_2), x_1, (x_2, x_3, x_2]}
N_{13}		$\mathbf{S_1}$ ={[(x_1, x_2, x_3, x_2, x_1), (x_2, x_3 , x_2), x_1, (x , x_3 , x_2)]}
N_{14}		$\mathbf{S_1}$ = {[x_1, x_2), (x_3, x_2, x_1, x_2, x_3), (x_2, x_1 , x_2), x_3 , (x_2]}
N_{15}		$\mathbf{S_1}$ = {[(x_1, x_2, x_3), (x_2, x_1, x_2, x_3, x_2), (x_1, x_2, x_3), x_2]}
N_{16}	Bounded four-sided subshapes	$\mathbf{S_1}$ = {3[S_2], 2[S_3]}
N_{17}		$\mathbf{S_1}$ = {3[S_2], [S_4]}
N_{18}	Bounded four-sided shapes & three-sided subshapes	$\mathbf{S_1}$ = {3[S_2], 2[S_5], 2[S_6]}
N_{19}		$\mathbf{S_1}$ = {3[S_2], 2[S_6], 2[S_7]}
N_{20}	Emergent shapes	$\mathbf{S_1}$ = {3[S_{14}], 3[S_{15}], [S_{16}]}
N_{21}		$\mathbf{S_1}$ = {3[S_{14}], 3[S_{18}], 6[S_{17}], [S_{16}]}
N_{22}		$\mathbf{S_1}$ = {3[S_{14}], 3[S_{18}], [S_{19}]}
N_{23}		$\mathbf{S_1}$ = {3[S_{14}], [S_{20}]}
N_{24}		$\mathbf{S_1}$ = {3[S_{21}], 3[S_{18}], [S_{16}]}
N_{25}		$\mathbf{S_1}$ = {[S_{22}], 3[S_{18}]}
N_{26}		$\mathbf{S_1}$ = {6[S_{23}], [S_{24}]}
N_{27}	Figure and ground	$\mathbf{S_1}$ = {[S_1], 3[S_{13}]}
N_{28}		$\mathbf{S_1}$ = {[S_1], 3[S_{12}]}

3.3 The Role of Multiple Representations in Situated Learning in Designing

Multiple representations aid in structuring the problem space because they introduce linked states and operators. The multiple representations approach is a constructive device that is essential in indeterministic and situated processes such as designing. The main difference between well-structured and the indeterministic nature of ill-structured problems is described in terms of the problem space and the degree to which operators are specified. In an ill-structured problem, the route to the goal state must be discovered,

while the goal state itself is not entirely clear. The use of multiple representations within learning environments could have many advantages (Ainsworth et al., 1996). Two broad classes of claims can be distinguished concerning the advantages of learning with multiple representations. The first is that they support different ideas and processes. Multiple representations can be used to place different emphases on aspects of complex ideas, and may be useful where one representation would be insufficient to carry all the intended information about the domain. Multiple representations support individual differences in representational and strategic preference. The second is that they promote a thorough comprehension of the domain (Ainsworth et al., 1996). On the other hand, different ways of representing the same information encourage or invite different ways of reasoning. Multiple representations allow for different interpretations by designers. They are commonly called perspectives in knowledge representation languages, views in the database and representations in the design environment. Multiple representations allow the coexistence of several descriptions of the same entities (Nguyen and Rieu, 1991)

The use of multiple representations in designing allows for the transformations of a design between design spaces. The interesting and intriguing point of using the multiple representations is that we never know in advance what information we will get out of a representation because of the richness of the design environment and the relationships between the representations and their meanings. From such multiple knowledge representations, such as spatial knowledge which could be used to represent shape and space in several ways, a diverse range of interpretations about the shape could be developed and different relationships among recognised knowledge may be encountered. A learning system may benefit from the divergences among the multiple representations as different states of situatedness where design knowledge was recognised. What makes multiple representations interesting in the context of situatedness is that they provide the opportunity for different and rich relationships to be constructed from what looks to be a single object. This allows a learning system to move through a number of representations (states of situatedness), in which the system can distinguish the situatedness of knowledge as it is being acquired.

Multiple representations benefit learners to the extent to which they recognise the regularities of the relationships among design knowledge across the observations constructed from the representations. In these representations relationships assume a central role. For example, in architectural plans, the individual components are usually not as interesting as the relationships among them. Multiple representations through re-representing designs from different views provide a platform to learn rich relationships between design knowledge (focus) recognised within these representations and the states of situatedness of these representations. The relationships to be learned include where this knowledge (focus) was operating and applicable. The regularities of these relationships help to construct the situatedness of that knowledge and have the potential to guide the use of knowledge when similar situations arise. In other words, multiple representations provide a platform for the learning system to construct the situatedness of knowledge.

It appears that humans have no difficulty in using different representations for what is apparently the same object in order to achieve different goals. The emergence of

patterns in the re-interpretations of designs helps in making some shape semantics recognisable. The re-interpretations contribute to making changes in a design environment and consequently recognising shape semantics that were not explicitly recognisable in the previous representations. Multiple representations of design through re-interpretation are proposed to serve as a platform for SLiDe. Multiple representations provide the opportunity for different shape semantics and relationships among them to be found from the shape 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.

Chapter 4

Situated Learning of Architectural Shape Semantics

"I am I plus my surroundings and if I do not preserve the latter, I do not preserve myself" Jose Ortega Y Gasset (Akman, 1999)

This Chapter addresses situated learning of architectural shape semantics. The chapter commences with the recognition of shape semantics in architectural design compositions (in the form of floor plans), because the formation and discovery of relationships among parts of the compositions are fundamental tasks in designing. Three sets of architectural shape semantics have been selected to be recognised from architectural drawings: expression, symmetry and modality. Each set includes a group of shape semantics. The recognition processes of these semantics are introduced. Multiple representations provide a platform for recognising various shape semantics from each representation. Such a platform helps to construct a set of observations from the representations. The recognition of shape semantics in architectural design compositions is useful but what is more useful is to learn the situatedness of these semantics within which they were recognised. The domain of shape semantics is used as vehicle to demonstrate the concept of situated learning in designing, however the underlying conceptual approach is applicable in other domains. The regularities of the relationships among shape semantics constructed from the observations of the design environment where shape semantics were recognised are the triggers for constructing the situatedness of these semantics. The situatedness of a shape semantic carries with it the applicability conditions of that shape semantic.

4.1 Shape Semantics in Architectural Drawings

Drawing is described as a representational medium or as a communicative tool used during the design process. More recently, the drawing itself and the way of seeing it have been explored as indispensable parts of the process of designing and the underlying design thinking (Liu, 1995; Suwa et al., 1998b). In architectural designing, as in many other design disciplines, shape composition is an important design activity. Shapes are the way we begin to understand the visual world that our visual sense brings to us (Marr, 1982). Through shapes designers express ideas and represent elements of design, abstract concepts and construct situations. Hence, their role in designing is significant. The formation and discovery of relationships among parts of a composition are fundamental tasks in designing (Mitchell and McCullough, 1995; Kolarevic, 1997). One of the analyses of architectural shapes in a drawing is the result of certain

relationships among its parts which characterise each and every design. The abstraction and explicitness of these relationships in a recognised drawing can therefore lead to a closer and better understanding of shape semantics (Koutamanis and Mitossi, 1993).

Shape semantics have many characteristics, one of which is they encapsulate design knowledge that can be ascribed to design artefacts and are among design knowledge that tend to be fundamental to aesthetic design. Shape semantics are the interpretation of visual patterns or visual forms of groups of shapes in the drawing (Jun, 1997). 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. Various types of shape semantics can be explained in a variety of ways by grouping structures using the laws of figure perception (Arnheim, 1977; Meiss, 1991). Grouping structures is supported by such factors as: repetition, similarity, proximity and orientation. Gestalt theory deals with the grouping phenomenon in a comprehensive way. The central concept of the theory is the concept of Gestalt-form or configuration of any segregated whole or unit (Köhler, 1970).

Shape semantics can be recognised in all architectural drawings. There are two types of shape semantics of concern in this thesis: primary and emergent. A primary shape semantic is a visual pattern of relationships among shapes in the initial representation. An emergent shape semantic is a visual pattern of relationships among shapes that exist only implicitly in the initial representation but has been made explicit within emergent shapes (Gero et al., 1995). Shape semantics could be discovered from a whole drawing or parts of a drawing. These semantics may play a crucial role in developing further ideas in the same design if designers pursue them further.

4.1.1 Selection of shape semantics

There are various sets of shape semantics that could be recognised in architectural drawings. In this thesis, only three sets of shape semantics of architectural design have been selected: expression, symmetry and modality as shown in Figure 4.1. The reason for this selection is that they are among the most prominent semantics in architectural shapes. These shape semantics are concerned with the visual relationships among shapes in a design composition. Expression indicates the impression of a feature or a defined assemblage of features within the composition such as dominance and adjacency. Visual dominance reflects the effect of shape size and spatial location in relation to other shapes where adjacency reflects the contiguity among shapes in the design composition. Symmetry indicates harmony and conformity among shapes within the composition such as reflective symmetry, cyclic rotation, translational repetition and scaling. Modality includes the characteristics of how these parts in the design composition (shapes), are put together such as centrality, linearity and radiality. Some examples of shape semantics in architectural design compositions are shown in Figures 4.2 to 4.5.

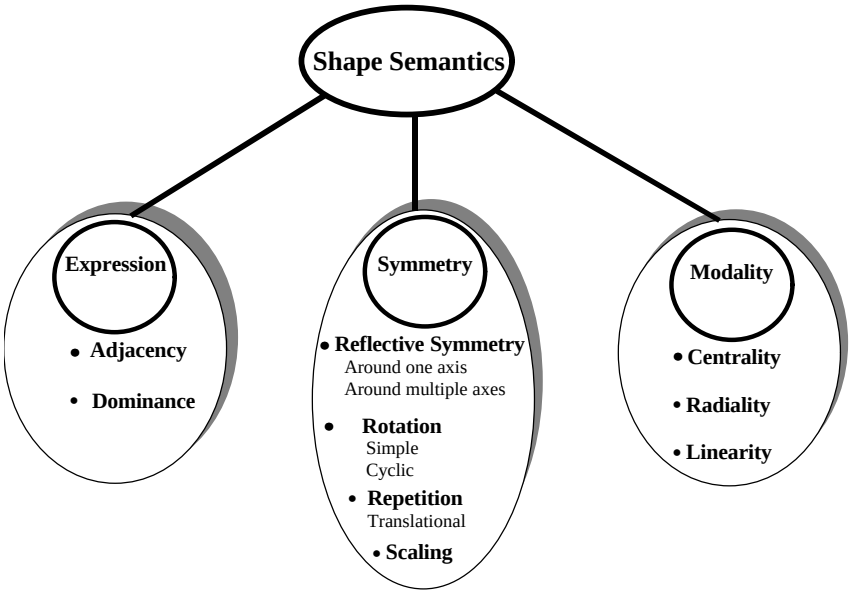


Figure 4.1 Three sets of shape semantics are selected to be recognised from architectural design compositions.

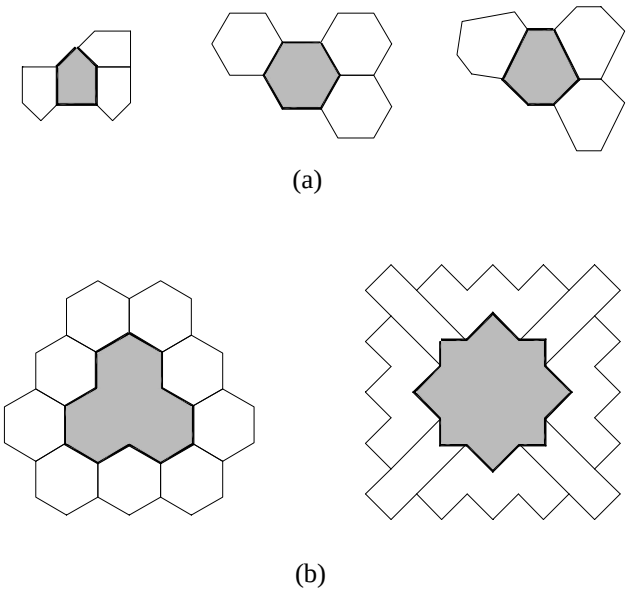


Figure 4.2 Examples of shape semantics representing architectural expressions among shapes within design compositions (a) adjacency and (b) dominance.

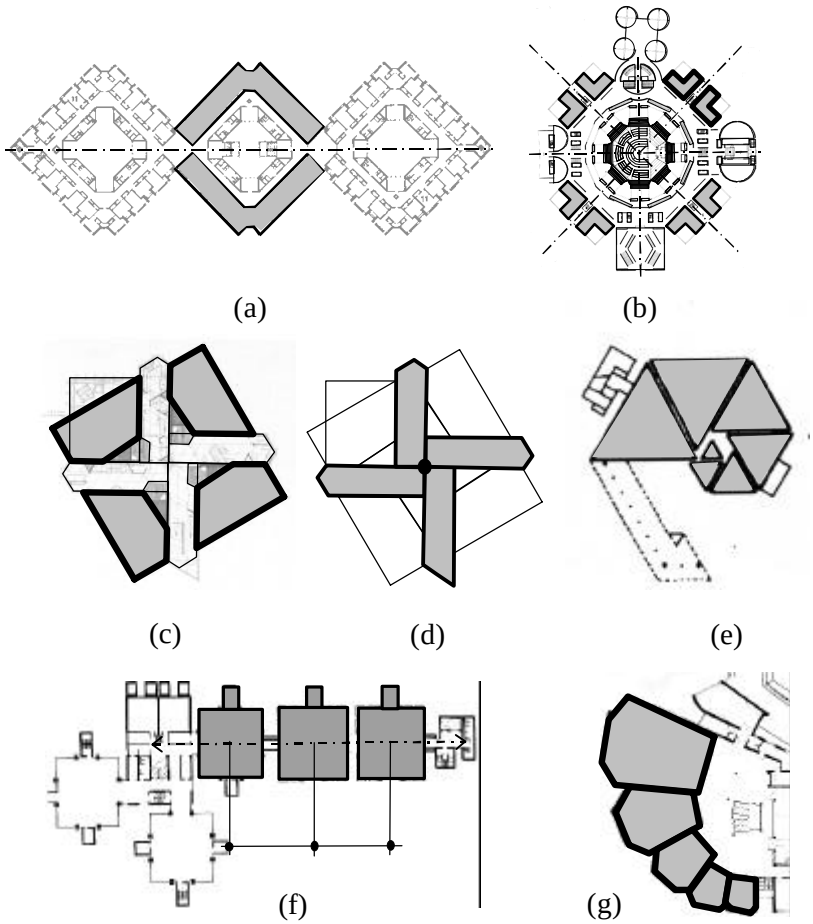


Figure 4.3 Examples of shape semantics representing congruence among parts of design compositions: (a) reflective symmetry around an axis, Erdman Hall Dormitory, Bryn Mawr by Louis I. Kahn; (b) reflective symmetry around multiple axes, National Assembly Hall in Dacca by Louis I. Kahn; (c) and (d) closed cyclic rotation, Price Tower, Bartlesville by Frank Lloyd Wright; (e) scaling, Holy Trinity Ukrainian Church by Radoslav Zuk; (f) translational repetition, Richards Medical Research Building, Philadelphia by Louis I. Khan; and (g) scaling, Wolfsburg Cultural Centre by Alvar Aalto.

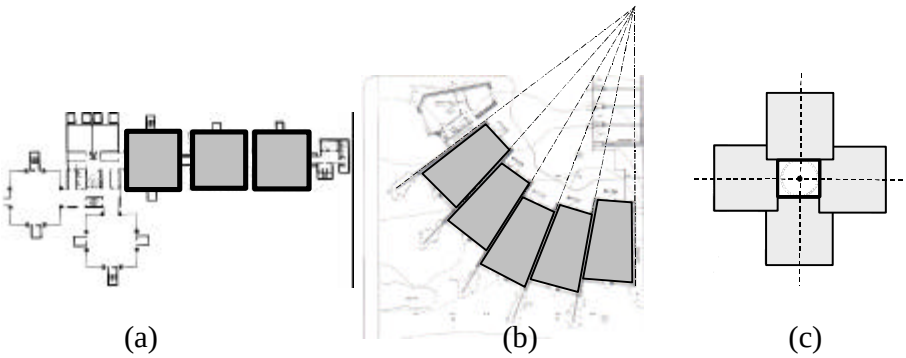


Figure 4.4 Examples of shape semantics representing the enclosure among shapes: (a) linearity; (b) radiality, Row house in Jakobstad by Alvar Aalto; and (c) centrality, Trenton Bath House by Louis I. Khan.

4.1.2 Recognising various shape semantics from multiple representations

There is a vast range of possible architectural shape semantics which could be emerged. For instance, different representations as shown in Figure 4.5 allow for some shape semantics to be readily recognised such as reflective symmetry, cyclic rotations, dominance, multiple reflective symmetry, simple rotation and centrality as shown in Figures 4.5(a) to 4.5(f) respectively. Each representation helps in the recognition of certain shape semantics whereas it could not be readily recognised at other representations. For instance, dominance cannot be easily recognised in the representations shown in Figure 4.5 except in the representation shown in Figure 4.5(c) wherein it can be readily recognised

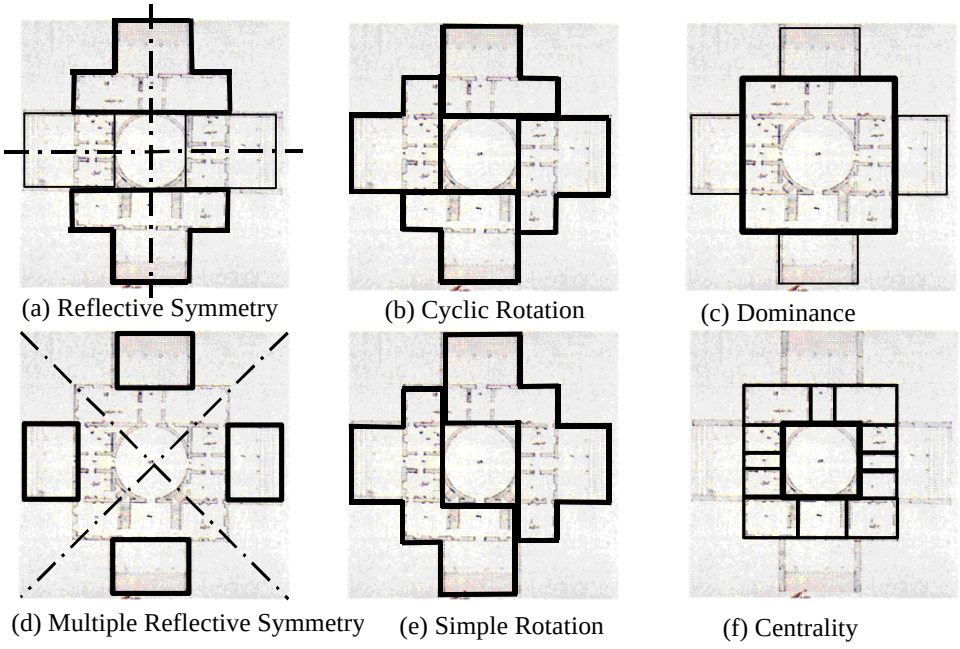


Figure 4.5 Recognition of different shape semantics from multiple representations of the same design composition.

4.2 Recognition of Shape Semantics

Shape semantics are recognised in terms of similarity of spatial relationships as well as physical properties. Shape semantics recognition starts from the 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, 1995a). 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. An example of two congruent shapes $\boldsymbol{S}_x$ and $\boldsymbol{S}_y$ is shown in Figure 4.5.

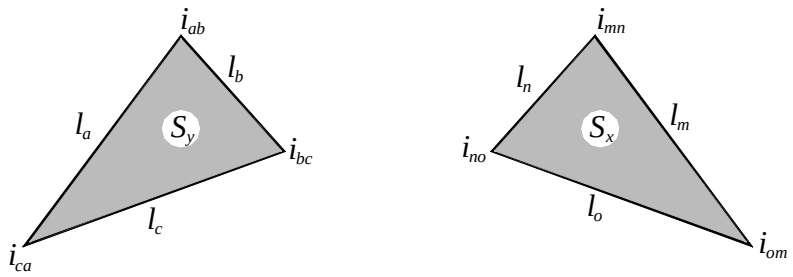


Figure 4.6 An example of two congruent shapes $\boldsymbol{S_x}$ and $\boldsymbol{S_y}$ where there are three vertices in each shape (i_{ab}, i_{bc} and i_{ca}) and (i_{mn}, i_{no} and i_{om}); the lengths of corresponding edges are equal, $l_a = l_m$, $l_b = l_n$ and $l_c = l_o$; the angles at corresponding vertices are equal; $A(l_a, l_b) = A(l_m, l_n)$, $A(l_b, l_c) = A(l_n, l_o)$ and $A(l_c, l_a) = A(l_o, l_m)$; and the ratios of each two consecutive edges are equal, $l_a \mid l_b = l_m \mid l_n$, $l_b \mid l_c = l_n \mid l_o$ and $l_c \mid l_a = l_o \mid l_m$.

4.2.1 Recognition of shape semantics indicating symmetry

4.2.1.1 Reflective symmetry around an axis

For reflective symmetry around an axis ($\boldsymbol{M_r}$) to exist, the slopes of line segments joining corresponding vertices of the two congruent shapes must be equal, ie such line segments must be parallel. The line joining the midpoints of the slopes (bisector) must be straight and perpendicular to them (March and Steadman, 1971; Baglivo and Graver, 1983). This line is the axis of reflection. An example of reflective symmetry around an axis is shown in Figure 4.7.

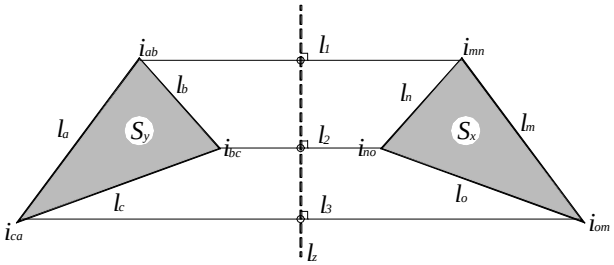


Figure 4.7 An example of reflective symmetry ($\boldsymbol{M_r}$) between two congruent shapes $\boldsymbol{S_x}$ and $\boldsymbol{S_y}$ around an axis l_z where $l_1 // l_2 // l_3$ and their slope is zero, l_z joining midpoints of l_1, l_2 and l_3 is a straight line, and $l_z \perp l_1$, $l_z \perp l_2$ and $l_z \perp l_3$.

4.2.1.2 Reflective symmetry around multiple axes

Reflective symmetry around multiple axes ($\boldsymbol{M_t}$) exists where at least four congruent shapes are available in a design composition. Each congruent shape has to be at least reflected around more than one axis. The two axes of reflection must be perpendicular on each other. An example of reflective symmetry around multiple axes is shown in Figure 4.8.

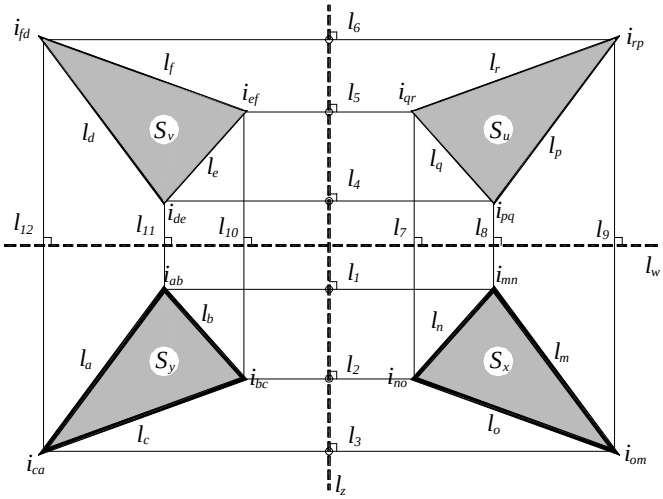


Figure 4.8 An example of multiple reflective symmetry (***M*****$_t$**) around two axes l_z and l_w where each of the congruent shapes ***S*****$_x$**, ***S*****$_y$**, ***S*****$_v$** and ***S*****$_u$** is reflected around both of them and $l_z \perp l_w$.

4.2.1.3 Simple rotation

For simple rotation (***R*****$_s$**) to exist, the perpendicular bisectors on the line segments joining the corresponding vertices of two congruent shapes must intersect at the same point (concurrent), (March and Steadman, 1971; Baglivo and Graver, 1983) as shown in Figure 4.9. The concurrent point is their rotational centre.

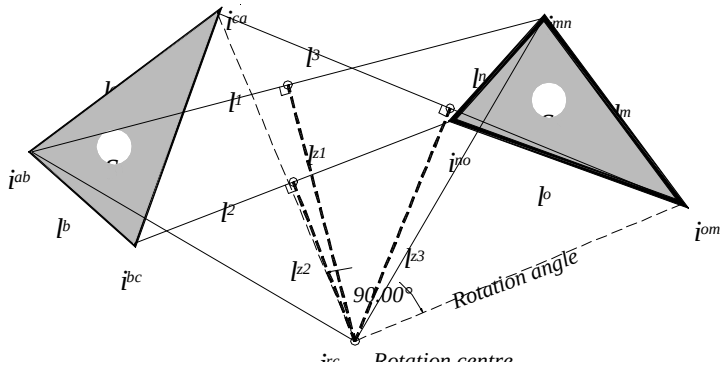


Figure 4.9 An example of simple rotation (***R*****$_s$**) between two congruent shapes ***S*****$_x$** and ***S*****$_y$** around a rotation centre point i_{rc}.

4.2.1.4 Cyclic rotation

Cyclic rotation (***R*****$_n$**) exists when at least three congruent shapes are available in a design composition around one rotational centre where the rotation angles between each two consecutive shapes are equal and have to be not more than 120^o. An example of cyclic rotation among four congruent shapes is shown in Figure 4.10.

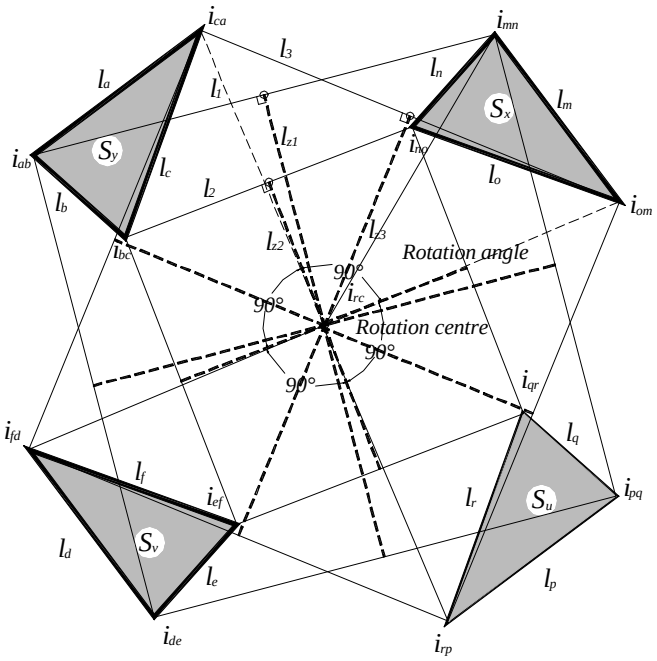


Figure 4.10 An example of cyclic rotation (***R**n*) between four congruent shapes ***S**x*, ***S**y*, ***S**v* and ***S**u* around the same rotational centre point i_{rc} with the same rotational angle 90°.

4.2.1.5 Translational repetition

Translational repetition (***P**r*) exists where at least three congruent shapes are available in the design composition where the line segments joining the corresponding vertices of each of two consecutive congruent shapes form part of a parallelogram. The other two sides of the parallelogram must be the corresponding sides of each of two congruent shapes (Baglivo and Graver, 1983). The distances from the centre of each of two consecutive congruent shape (translational distance), must be equal. Figure 4.11 shows an example of translational repetition among three congruent shapes.

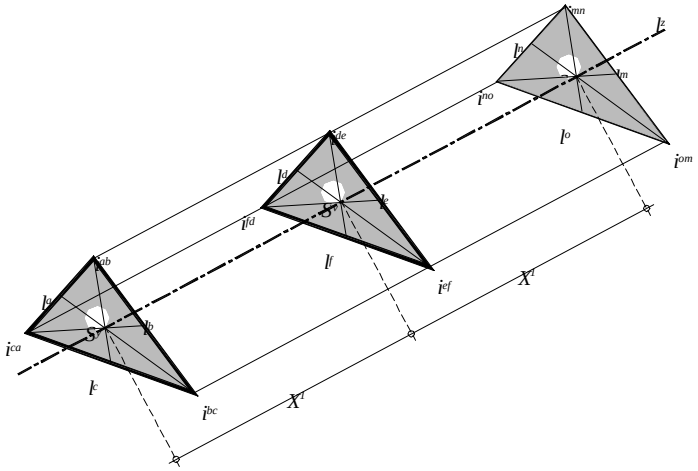


Figure 4.11 An example of translational repetition (***P**r*) between three congruent shapes ***S**x*, ***S**y* and ***S**v* around translational line l_z with the same translational distance X_1.

4.2.1.6 Scaling

Comparing all synchronised line segments of two shapes where their ratios are all equal identifies scaling (***E_s***) among shapes in the design composition. An example of scaling is shown in Figure 4.12.

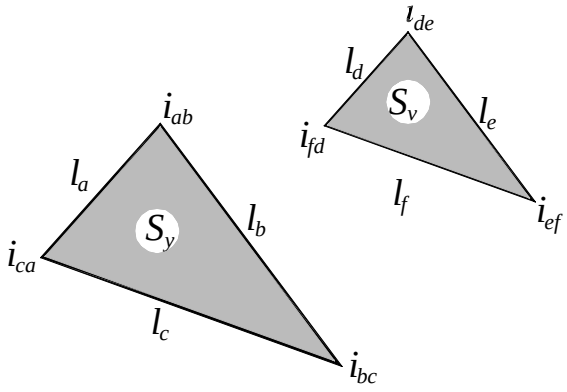


Figure 4.12 An example of scaling (***E_s***) between two shapes ***S_y*** and ***S_v*** where $l_a / l_d = l_b / l_e = l_c / l_f$.

4.2.2 Recognition of shape semantics indicating expression

4.2.2.1 Adjacency

Adjacency (***A_d***) is a boundary relationship among shapes in a design composition. In this thesis, neither shape overlap nor partial sharing is permitted during the recognition of shapes. Hence, adjacency exists when two shapes are contiguous. Two shapes are contiguous when they are discrete, ie their product is empty (Chase, 1997). Two shapes are adjacent if they share at least one edge of their boundaries. Examples of different kinds of adjacency are shown in Figure 4.13.

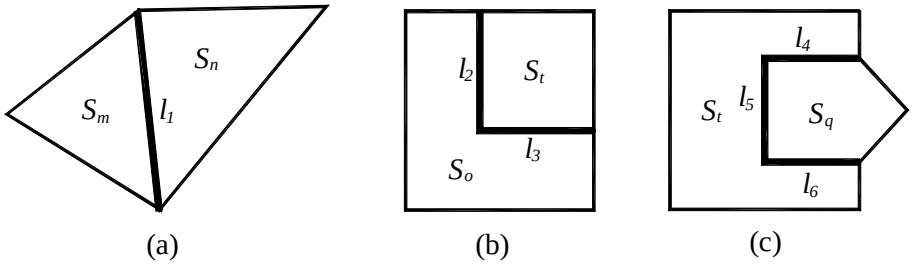


Figure 4.13 Examples of different kinds of adjacency (***A_d***) among shapes.

4.2.2.2 Dominance

Visual dominance (***D_m***) exists in a design composition if there is a pre-eminent shape in size that has an area at least one third of the total area in a design composition. Such a shape could be of contiguous or non-contiguous relationship to other shapes in the composition but the centre of the design composition must occur within the boundary of the dominant shape as shown in Figure 4.14.

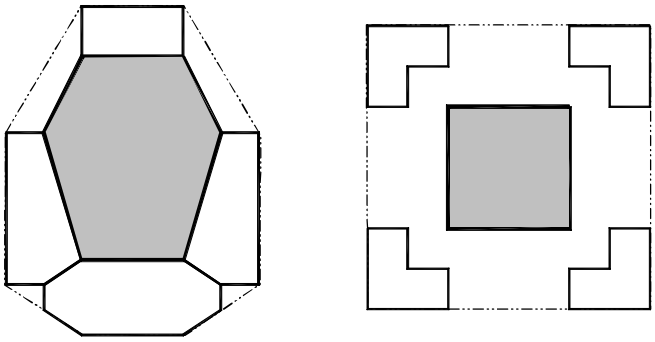


Figure 4.14 Examples of dominance (D_m) among contiguous and non-contiguous shapes in (a) and (b) respectively.

4.2.3 Recognition of shape semantics indicating modality

4.2.3.1 Centrality

Centrality (C_e) exists in a design composition if there is a shape within the design composition in which the centre of the whole design composition occurs within its boundary. This centre must be of equal distances to centres of the congruent shapes in the design composition. Some examples of centrality in design compositions are shown in Figure 4.15.

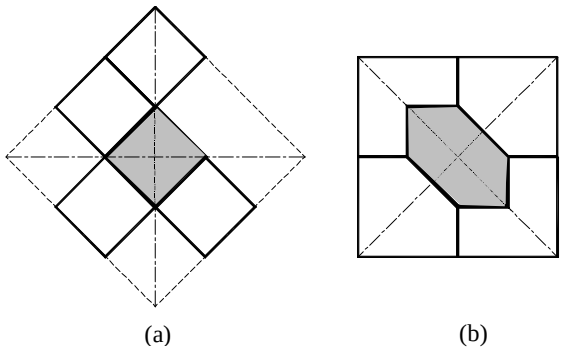


Figure 4.15 Examples of centrality (D_m) of a shape in some design compositions.

4.2.3.2 Radiality

Radiality (T_r) in a design composition exists where at least three shapes are rotated around a central point with different rotational angles. Cyclic rotation is a special case of radiality where any rotation angles between any two consecutive shapes have to be equal and not more than 120^o as shown in Figure 4.10. Examples of radiality among shapes are shown in Figure 4.16.

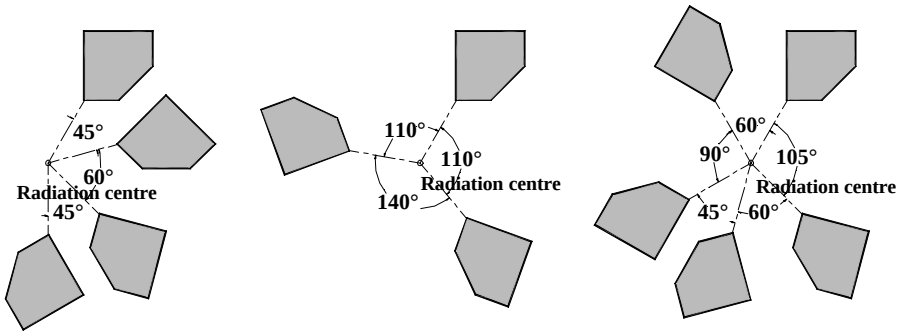


Figure 4.16 Examples of radiality (***T_r***) among congruent shapes.

4.2.3.3 Linearity

Linearity (***L_s***) exists in a design composition where at least three congruent shapes are repeated along a linear axis and the distances among the centres of consecutive congruent shapes are not equal. Translational repetition is a special case of linearity where the translational distances between each two consecutive congruent shapes are equal. Linearity exists as well among scaled congruent shapes as shown in Figure 4.17.

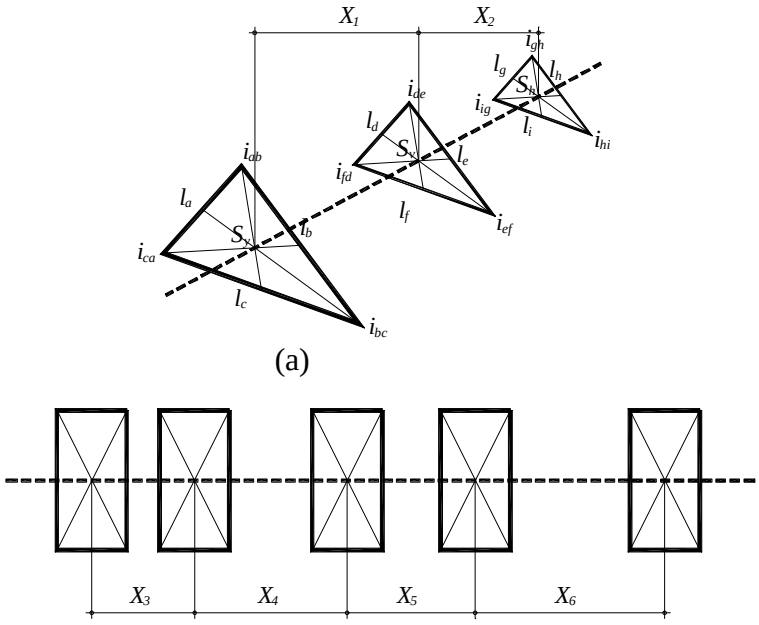


Figure 4.17 Examples of linearity (***L_s***) among (a) scaled congruent shapes or (b) congruent shapes.

4.3 Constructing Observations from Multiple Representations

Shape semantics can be recognised within each of the representations. The group of shape semantics recognised in each representation is considered as an observation constructed from that representation. Since various shape semantics are recognised from different representations, multiple representations (N_i) help in constructing a set of observations (O_j) of a design composition. This set of observations is used to search for regularities of relationships among shape semantics within which these shape semantics were recognised. The set of observations shown in Table 4.1 is constructed from the representations shown in Section 3.2.2. The notations of shape semantics within the observations in Table 4.1 are as follows:

A_d Adjacency
C_e Centrality
D_m Dominance
M_r Reflective symmetry around one axis
M_t Reflective symmetry around multiple axes
R_s Simple rotation
R_n Cyclic rotation

4.4 Situated Learning of Shape Semantics

The processes of locating shape semantics in relation to their situations within which they were recognised are as follows:

(i) select a single shape semantic and consider it as the knowledge in focus;
(ii) find all the observations within which this single shape semantic has been recognised;
(iii) find other shape semantics in these observations;
(iv) find the regularities of relationships among other shape semantics that are in conjunction with the knowledge in focus across the observation; such regularities carry the applicability conditions of where the knowledge in focus was recognised;
(v) construct the situatedness of the knowledge in focus based upon these regularities

4.4.1 Constructing the situatedness of shape semantics

Assuming that a shape semantic is labelled k_i when it is selected to be the knowledge in focus, it will be referred to as F_j and its learned situation as t_n. When applying the above procedures to the set of observations shown in Table 4.1, the following can be found. If k_1 refers to centrality (C_e) is chosen to be the knowledge in focus F_1, it is found that it has been recognised in different observations: O_{17}, O_{20}, O_{21}, O_{22}, O_{23}, O_{24} and O_{27}. Across these observations, two kinds of regularities were found as shown in Figure 4.18. These regularities are the mapping of one to many, one focus to many situations. The first regularity is found in the observations O_{17}, O_{22}, O_{23} and O_{27}.

Table 4.1 A set of observations constructed from the multiple representations developed from a design compositions as shown in Section 3.2.2.

Rep. No.	Obs. No.	Observation	Rep. No.	Obs. No.	Observation
N1	***O1***	$\boldsymbol{R_n}(x_1, x_2)$ $\boldsymbol{R_n}(x_3, x_2)$	***N2***	***O2***	$\boldsymbol{R_n}(x_2, x_3)$ $\boldsymbol{R_n}(x_2, x_1)$
N3	***O3***	$\boldsymbol{R_n}(x_1, x_2, x_3)$	***N4***	***O4***	$\boldsymbol{R_n}(x_3, x_2, x_1)$
N5	***O5***	$\boldsymbol{R_n}(x_2, x_3, x_2)$ $\boldsymbol{M_r}(x_2, x_3, x_2)$	***N6***	***O6***	$\boldsymbol{R_n}(x_2, x_1, x_2)$ $\boldsymbol{M_r}(x_2, x_1, x_2)$
N7	***O7***	$\boldsymbol{R_n}(x_1, x_2, x_3, x_2)$	***N8***	***O8***	$\boldsymbol{R_n}(x_2, x_3, x_2, x_1)$
N9	***O9***	$\boldsymbol{R_n}(x_3, x_2, x_1, x_2)$	***N10***	***O10***	$\boldsymbol{R_n}(x_2, x_1, x_2, x_3)$
N11	***O11***	$\boldsymbol{M_r}(x_2, x_3, x_2, x_1, x_2)$	***N12***	***O12***	$\boldsymbol{M_r}(x_2, x_1, x_2, x_3, x_2)$
N13	***O13***	$\boldsymbol{M_r}(x_2, x_3, x_2)$	***N14***	***O14***	$\boldsymbol{M_r}(x_2, x_1, x_2)$
N15	***O15***	$\boldsymbol{R_s}(x_1, x_2, x_3)$	***N16***	***O16***	$\boldsymbol{M_r}[S_2]$, $\boldsymbol{M_r}[S_3]$, $\boldsymbol{R_n}[S_2]$, $\boldsymbol{A_d}[S_3] \mid 2\,[S_2]$, $\boldsymbol{A_d}[S_3] \mid [S_3]$
N17	***O17***	$\boldsymbol{M_r}[S_2]$, $\boldsymbol{R_n}[S_2]$, $\boldsymbol{A_d}[S_4] \mid 3\,[S_2]$, $\boldsymbol{C_e}[S_4]$, $\boldsymbol{D_m}[S_4]$	***N18***	***O18***	$\boldsymbol{M_r}[S_2]$, $\boldsymbol{M_r}[S_6]$, $\boldsymbol{R_n}[S_2]$, $\boldsymbol{A_d}[S_2] \mid [S_6]$, $\boldsymbol{A_d}[S_5] \mid [S_6]$
N19	***O19***	$\boldsymbol{M_r}[S_2]$, $\boldsymbol{M_r}[S_6]$, $\boldsymbol{M_r}[S_7]$, $\boldsymbol{R_n}[S_2]$, $\boldsymbol{A_d}[S_2] \mid [S_6]$, $\boldsymbol{A_d}[S_2] \mid [S_7]$, $\boldsymbol{A_d}[S_6] \mid [S_7]$	***N20***	***O20***	$\boldsymbol{M_r}[S_{14}]$, $\boldsymbol{M_r}[S_{15}]$, $\boldsymbol{R_n}[S_{14}]$, $\boldsymbol{A_d}[S_{16}] \mid 3\,[S_{15}]$, $\boldsymbol{A_d}[S_{15}] \mid 2\,[S_{14}]$, $\boldsymbol{C_e}[S_{16}]$
N21	***O21***	$\boldsymbol{M_r}[S_{14}]$, $\boldsymbol{M_r}[S_{17}]$, $\boldsymbol{M_r}[S_{18}]$, $\boldsymbol{R_n}[S_{14}]$, $\boldsymbol{R_n}[S_{17}]$, $\boldsymbol{R_n}[S_{18}]$, $\boldsymbol{A_d}[S_{16}] \mid 6\,[S_{17}]$, $\boldsymbol{A_d}[S_{14}] \mid 2\,[S_{17}]$, $\boldsymbol{A_d}[S_{18}] \mid 2\,[S_{17}]$ $\boldsymbol{C_e}[S_{16}]$,	***N22***	***O22***	$\boldsymbol{M_r}[S_{14}]$, $\boldsymbol{M_r}[S_{18}]$, $\boldsymbol{R_n}[S_{14}]$, $\boldsymbol{R_n}[S_{18}]$, $\boldsymbol{A_d}[S_{19}] \mid 3\,[S_{14}] \mid 3\,[S_{18}]$, $\boldsymbol{C_e}[S_{19}]$, $\boldsymbol{D_m}[S_{19}]$
N23	***O23***	$\boldsymbol{M_r}[S_{14}]$, $\boldsymbol{R_n}[S_{14}]$, $\boldsymbol{A_d}[S_{20}] \mid 3\,[S_{14}]$, $\boldsymbol{C_e}[S_{20}]$ $\boldsymbol{D_m}[S_{20}]$	***N24***	***O24***	$\boldsymbol{M_r}[S_{18}]$, $\boldsymbol{M_r}[S_{21}]$ $\boldsymbol{R_n}[S_{18}]$, $\boldsymbol{R_n}[S_{21}]$ $\boldsymbol{A_d}[S_{16}] \mid 3\,[S_{21}]$, $\boldsymbol{A_d}[S_{18}] \mid 2\,[S_{21}]$, $\boldsymbol{C_e}[S_{16}]$
N25	***O25***	$\boldsymbol{M_r}[S_{18}]$, $\boldsymbol{R_n}[S_{18}]$, $\boldsymbol{A_d}[S_{22}] \mid 3\,[S_{18}]$, $\boldsymbol{D_m}[S_{22}]$,	***N26***	***O26***	$\boldsymbol{M_r}[S_{23}]$, $\boldsymbol{R_n}[S_{23}]$, $\boldsymbol{A_d}[S_{24}] \mid 6\,[S_{23}]$, $\boldsymbol{D_m}[S_{24}]$
N27	***O27***	$\boldsymbol{M_r}[S_{13}]$, $\boldsymbol{R_n}[S_1]$, $\boldsymbol{A_d}[S_1] \mid 3\,[S_{13}]$, $\boldsymbol{C_e}[S_1]$, $\boldsymbol{D_m}[S_1]$	***N28***	***O28***	$\boldsymbol{M_r}[S_{12}]$, $\boldsymbol{R_n}[S_{12}]$, $\boldsymbol{A_d}[S_1] \mid 3\,[S_{12}]$, $\boldsymbol{D_m}[S_1]$

The second regularity is found in the observations O_{20}, O_{21} and O_{24} . Across the observations O_{17}, O_{22}, O_{23} and O_{27} , when F_1 is selected the knowledge in focus, there is regularity of other shape semantics k_2, k_3, k_4 and k_5 that refer to reflective symmetry around an axis (M_r), cyclic rotation (R_n), adjacency (A_d) and dominance (D_m) respectively. This regularity defines the relationships under which F_1 is recognised and constructs the situation t_1 for the knowledge in focus F_1. This means that F_1 is situated within the shape semantics k_2, k_3, k_4 and k_5. In Figure 4.18, the unshaded curved arrow refers to the knowledge in focus and the shaded curved arrow refers to the situation of that focus. Table 4.2 illustrates this relationship between the selected knowledge in focus F_1 and its situation t_1.

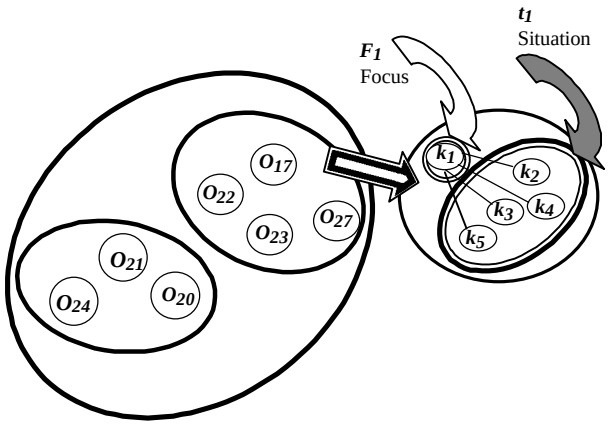


Figure 4.18 A learned regularity across the observations O_{17}, O_{22}, O_{23} and O_{27}. If k_1, centrality (C_e) is chosen to be the knowledge in focus, then k_2, k_3, k_4 and k_5 form its situation t_1.

Table 4.2 An example of centrality (C_e) as knowledge in focus and its situation across the set of observations.

O_i	**Focus** (F_1)	**Situation** (t_1)
O_{17} O_{22} O_{23} O_{27}	k_1, centrality (C_e)	k_2, reflective symmetry around an axis (M_r) k_3 cyclic Rotation of (R_n) k_4 adjacency (A_d) k_5 dominance (D_m)

4.4.2 Duality between knowledge in focus and the situation

Alternatively taking another shape semantic k_5, which refers to dominance (D_m), to be the knowledge in focus F_5 within the same regularity, it is found that k_1, k_2, k_3 and k_4 construct the situation t_5 of F_5 within which it was recognised as shown in Table 4.3 and Figure 4.19. In Figure 4.19, the inverted arrow refers to the duality between focus parts within the regularity. This could be explained as a duality between knowledge in focus and its situation within the same regularity. It means that, for certain knowledge in focus F_1 recognised in relation to the situation t_1 wherein k_5 is part of that situation, it is

possible that when k_5 is chosen to be the knowledge in focus F_5 its situation t_5 includes k_1 as one of its parts.

Table 4.3 An example of the duality between knowledge in focus and parts of its situation within the same regularity.

O_i	**Focus** (F_5)	**Situation** (t_5)
O_{17} O_{22} O_{23} O_{27}	k_5 dominance (D_m)	k_1, centrality (C_e) k_2, reflective symmetry around an axis (M_r) k_3 cyclic Rotation of (R_n) k_4 adjacency (A_d)

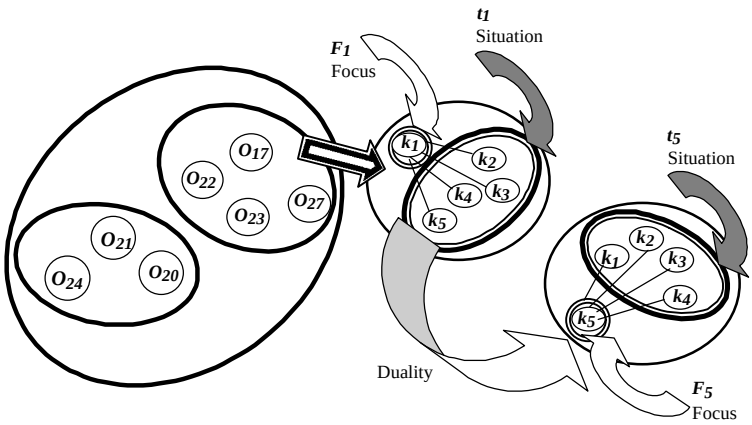


Figure 4.19 An example of the duality between knowledge in focus and parts of its situation within the same regularity.

In the second regularity learned across the observations O_{20}, O_{21} and O_{24}, if k_1 that refers to centrality (C_e) is chosen to be the knowledge in focus F_1, it is found that the other shape semantics k_2, k_3 and k_4 construct another possible situation t_{101} within which centrality (C_e) is recognised. It can be noticed that k_5 was in conjunction with F_1 at the first regularity but such a relationship does not appear to be seen within the second regularity. Applying the notion of duality within this regularity when k_5 which refers to dominance (D_m) is selected to be the knowledge in focus F_5, then its situation will be t_5 which is constructed from k_2, k_3 and k_4 as shown in Figure 4.20.

Applying the notion of duality can help with constructing the situatedness of other recognised shape semantics within the learned regularities. For instance, Figure 4.21 shows two of the possible situations t_2 and t_{201} of k_2 which refers to reflective symmetry (M_r) around one axis when it is considered to be the knowledge in focus F_2.

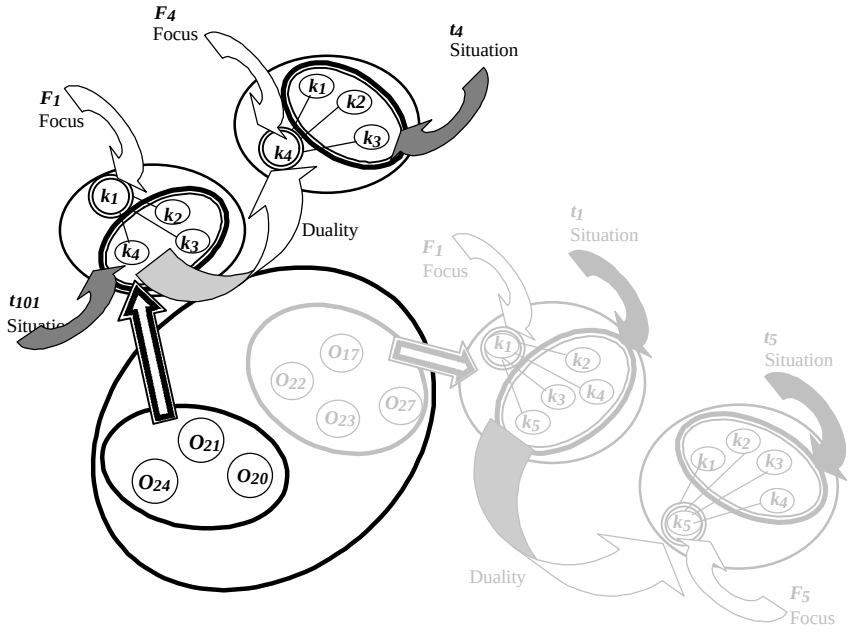


Figure 4.20 Another possible situation of k_1, which refers to centrality (C_e) constructed from the second regularity when it is considered as the knowledge in focus F_1. The duality between knowledge in focus F_4 and its situation within this regularity is shown. In this Figure, Figure 4.19 is illustrated in a dropped tone.

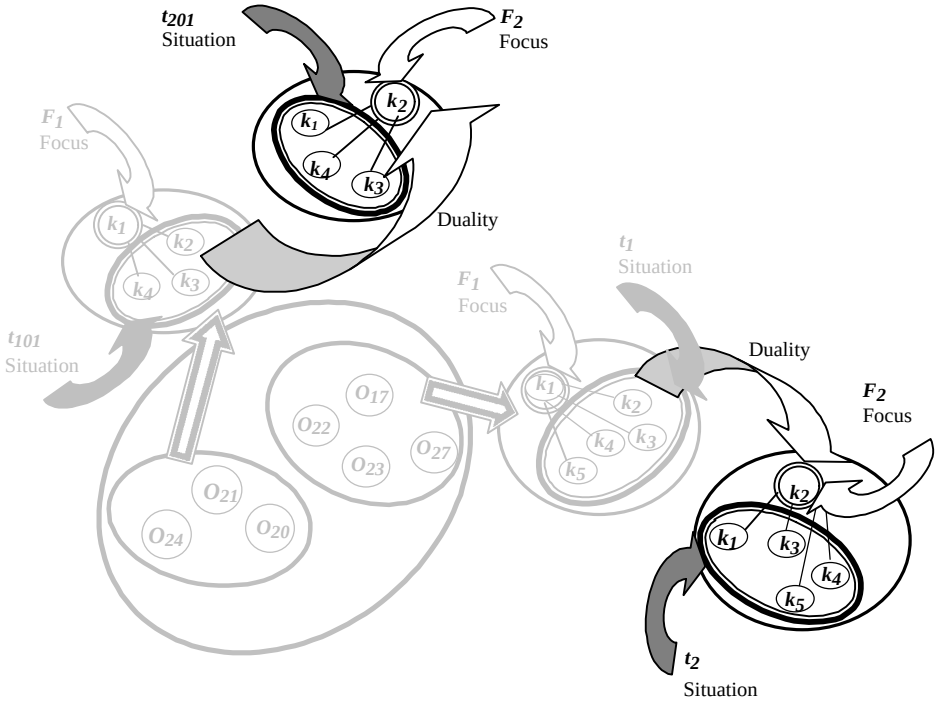


Figure 4.21 An example of applying the duality to construct two of the possible situations, t_2 and t_{201}, of reflective symmetry (M_r) around one axis within the two learned regularities.

4.4.3 Learning multiple situations for a certain knowledge in focus

Similarly, duality can be applied to learn the situations of other shape semantics recognised within those two learned regularities. However, in order to learn other possible situations in relation to certain knowledge in focus, the set of observations shown in Table 4.1 can be searched for other regularities within which that knowledge in focus is found. For instance, the set of observations shown in Table 4.1 can be searched for regularities within which k_3 (refers to cyclic rotation (R_n)) is recognised to be the knowledge in focus F_3. Four regularities are found within which there are four possible situations, t_3, t_{301}, t_{302} and t_{3o3}, within which F_3 is operating as shown in Figure 4.22. The same process could be used to locate every other shape semantic in relation to situations within which it was recognised across the set of observations constructed from the developed representations. This shows the importance of the regularities of relationships among shape semantics to construct their situations within which they were recognised. The situatedness of design knowledge implies its applicability conditions may serve as a basis for guiding its use when similar situations arise.

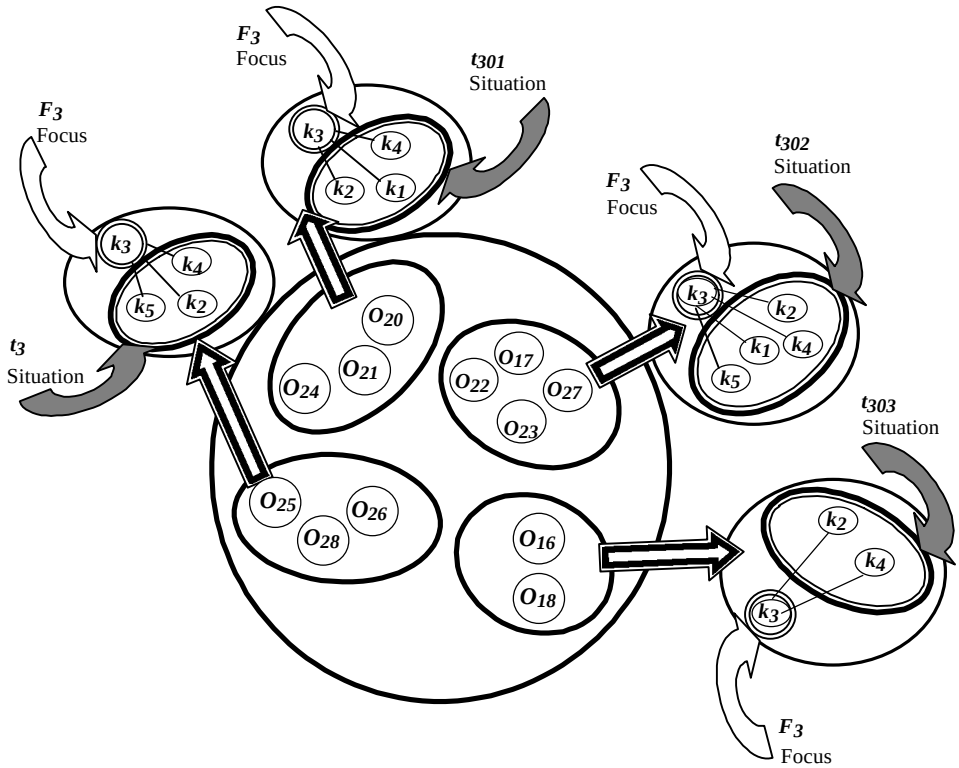


Figure 4.22 Four regularities are found across the set of observations shown in Table 4.1 within which the situations t_3, t_{301}, t_{302} and t_{3o3} are constructed in relation to k_3 which refers to cyclic rotation (R_n) when considered as the knowledge in focus.

4.4.4 Preconditions vs. Situatedness of Shape Semantics

Each shape semantic has preconditions without which it cannot be recognised. For instance, the preconditions of centrality (C_e) to be recognised in a design composition are: the centre of the whole design composition must occur within the boundary of a shape within the design composition; and this centre must be of equal distances to centres of the congruent shapes in the design composition as shown by example in Section 4.2.3.1. These preconditions are necessary and sufficient conditions for centrality (C_e) to be recognised. At the same time, the preconditions indicate nothing about the situation within which centrality (C_e) could be recognised. Centrality (C_e) operates within unique situations in the design environment. The situatedness of centrality (C_e) determines its applicability conditions within the design environment. The primary difference between the preconditions and the situatedness is that the latter cannot be completely predetermined but has to be constructed. For instance, the situatedness of centrality (C_e) is constructed based on the regularities of relationships between centrality (C_e) and other shape semantics in the design environment within which centrality (C_e) was recognised. The regularities of relationships learned from a set of observations of a design composition and shown in Figure 4.18 and Table 4.2 indicated that centrality (C_e) is situated within reflective symmetry around an axis (M_r), cyclic rotation (R_n), adjacency (A_d) and dominance (D_m). In other words, the difference between preconditions and situatedness of design knowledge (shape semantics) is that the former is fixed and the latter is changing based on what is constructed from the design environment.

Chapter 5

A Computational System for Situated Learning in Designing (SLiDe)

This Chapter presents a computational system for situated learning in designing (SLiDe). SLiDe is implemented to operate within the domain of architectural shape semantics. Its underlying concepts could be used in other domains. The Chapter commences by introducing the framework of SLiDe. SLiDe consists of three primary modules: Generator, Recogniser and Incremental Situator. The Generator is used by the designer to develop a set of multiple representations of a design composition. This set of representations form the initial design environment of SLiDe. The Recogniser detects shape semantics within the representations and produces a set of observations, each of which consists of a group of shape semantics recognised at each representation. The Incremental Situator consists of two sub-modules: Situator and Restructuring Situator. The Situator module locates shape semantics in relation to the situations within which they were recognised by finding the regularities of relationships among them across the observations and clustering them into situational categories organised in a hierarchical tree structure. Such relationships change over time due to the fluid nature of designing that causes changes in the design environment due to the development of further representations of other ways of viewing the same design composition. The Restructuring Situator updates the previously learned situational categories and restructures the hierarchical tree accordingly. An illustration of how the Incremental Situator works is presented.

5.1 Framework for Situated Learning in Designing

The role of the computational system for **S**ituated **L**earning **i**n **De**sign (**SLiDe**) is to locate design knowledge, in the form of shape semantics, in relation to their situations within which they were recognised within the design environment, ie. learning the applicability conditions of design knowledge. This is achieved by learning the regularities, in the form of situational categories, of relevant relationships among the shape semantics across different observations constructed from the set of multiple representations. Developing multiple representations from a single design composition was introduced in Chapter 2, and methods of shape semantic recognition and constructing situational categories from the set of observations were discussed in Chapter 4. SLiDe is an integrated system for generating multiple representations, shape semantics recognition, constructing observations and clustering situational categories of

shape semantics over time within which they were recognised. SLiDe takes a design composition in the form of line segments as its input. It generates, with the interaction of the designer, a set of multiple representations of that design composition which serves as its initial design environment and constructs a set of observations through the process of shape semantics recognition from each representation. SLiDe produces, in a hierarchical structure, a set of situational categories within which shape semantics were recognised in the design environment. The designer may choose to pursue developing further representations that could be generated using SLiDe once again in addition to the previous representations. This causes a change in the design environment of SLiDe by adding new representations. Consequently, SLiDe updates its own observations from the design environment considering such changes. SLiDe restructures its previously learned situational categories without being affected by the order of observations and dynamically produces a new hierarchical structure of situational categories in response to the changes that took place in its design environment. This indicates that SLiDe modifies its behaviour in response to the changes that have taken place in its design environment, ie SLiDe situates its learned knowledge in relation to the design environment.

SLiDe's framework consists of three modules: Generator, Recogniser and Incremental Situator that includes Situator and Restructuring Situator as shown in Figure 5.1. The Generator module assists the designer in generating multiple representations of a single design composition, here in the form of a floor plan. These representations serve as a platform and form the design environment for the Recogniser module to interact with and construct its own set of observations from the design environment for the Situator module to learn from. The Recogniser module detects each representation and attributes shape semantics to it. The result of using the Recogniser module is a set of observations, each of which comprises a group of shape semantics associated with each corresponding representation. 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 in a hierarchical tree structure. The situational categories are clustered based on the regularities of relationships among shape semantics within which they were recognised across the observations. The reason for calling them situational categories rather than normal categories or clusters as is the case in machine learning is that knowledge is generalised with respect to the situation within which it was recognised. Generating a set of new representations of the same design decomposition triggers the Recogniser module to update its own set of observations from the design environment which consequently triggers the Restructuring Situator module to update its previously learned situational category and its hierarchical tree structure. 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 in the form of new constructed situations. The results of SLiDe are sets of situational categories that situate the shape semantics.

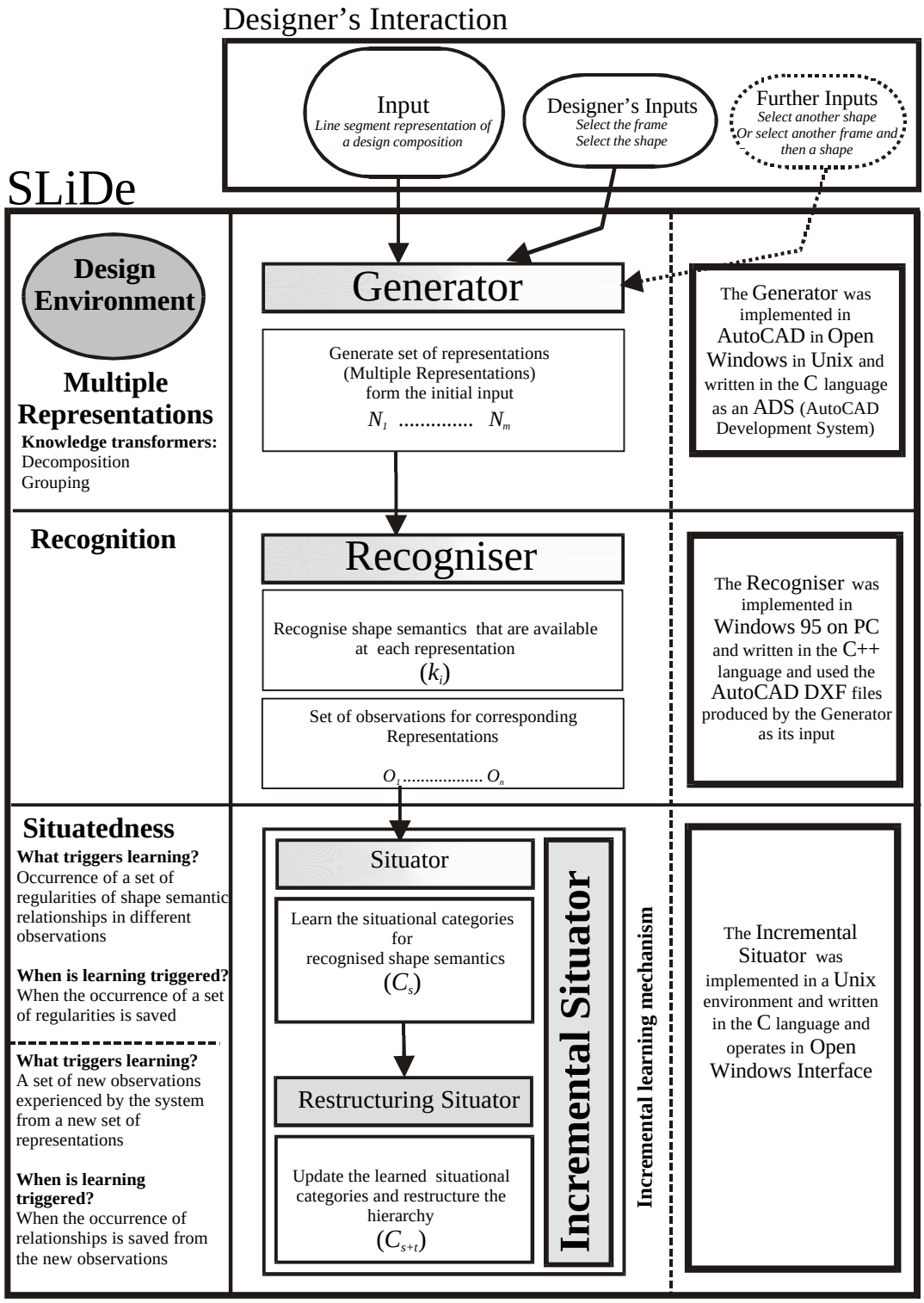


Figure 5.1 The overall framework of the computational system for situated learning in designing (SLiDe).

5.2 Multiple Representations using the Generator Module

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 5.2. The Generator module commences with an initial representation of a design composition, in the form of line segments, as its data input. It re-represents these line segments in the form of their infinite maximal lines within the frame selected by the designer. This is achieved by extending the line segments representing the shape to the boundary of that frame. The Generator requests the designer to select a shape of interest from among the intersections of the infinite maximal lines. The boundary of the selected frame defines the design space to be searched by the Generator module. The Generator searches the design space for a congruent shape of the selected shape and generates a representation from the combination of congruent shapes and the boundary of the initial representation. If there are no congruent shapes in the design space the representation is generated from the combination of the selected shape and the boundary of the initial representation. The Generator does not allow for overlapping shapes while searching for congruent shapes. This representation generation process is repeated with different selections to develop a set of multiple representations. The process is terminated when the designer is not interested in further selections at that moment of time.

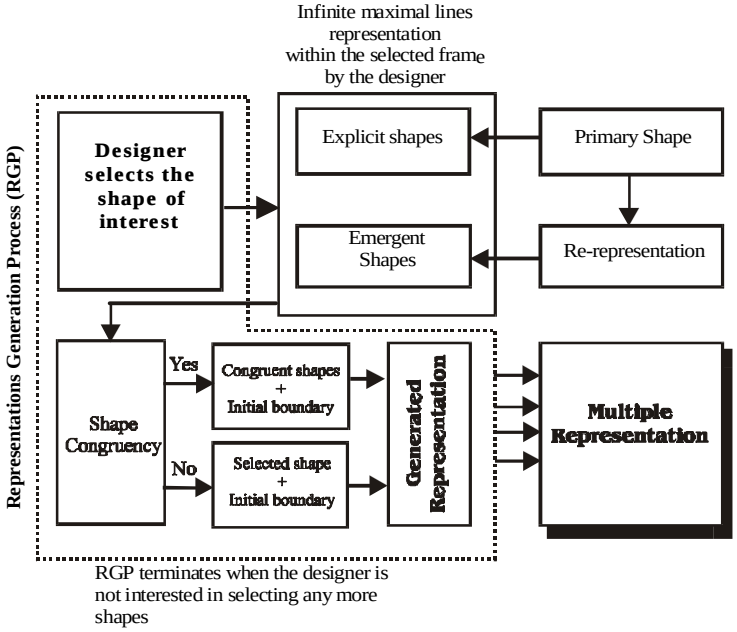


Figure 5.2 The Process model of the Generator module to develop multiple representations.

The size and location of the frame around the design composition affects the set of representations that can be generated since they affect the number of intersections of infinite maximal lines that can be generated. The designer may choose later to pursue developing other representations of the same design composition either by selecting other frames or other selections of shapes within the intersections of infinite maximal lines. The interaction between the designer and the Generator module provides a medium of communication between the designer and the system that affects the output of

developed representations based upon designer's interest and at the same time limits the search space. The selection of shapes among the intersections could be fully automated and computed randomly but there is no direct basis to select one shape rather than the other.

A commercially available CAD system has been adopted around which to develop the Generator module. For this purpose AutoCAD as an industry CAD system was selected. The Generator module has been implemented in AutoCAD in Open-Windows in a Unix environment and it is written in the C language and as an ADS (AutoCAD Development System) application. The Generator module include five sub-modules, some of which are based in part on SESS (Jun, 1997) and SPARS (Cha, 1998) and performed as follows:

draw a frame around the design composition
{
get the initial vertices;
draw a frame around the design composition;
correct the frame if it is not convenient;
get the entities within the frame;
}
infinite maximal lines representation
{
get the entities that are straight lines within the frame;
extend these lines to the frame;
get the intersections of these lines;
ignore the intersections of these lines with the boundary of the frame;
}
selection of a bounded shape (polyline)
{
get the intersections (vertices) of the shape selected by the designer;
get the distances (lengths) between each two consecutive vertices (edges);
get the number of edges;
get the angles at vertices (between each two consecutive edges);
}
shape congruency identification
{
find the corresponding first intersection (vertex) of the presumed congruent shape among all intersections in the frame
{
choose two consecutive line segments l_1 and l_2 at an intersection to be the first intersection of the selected shape;
get the distances d_1 and d_2 at an intersection i_1 within the frame that is equal to l_1 or l_2;
get the angle at the vertex i_1 between d_1 and d_2;
compare the angles on vertices between d_1 and d_2 and l_1 and l_2 (must be equal to continue);
get the i_1 to be the first intersection and the other endpoint of d_1 to be the second intersection;
get d_1 to be the corresponding line segment to l_1 and d_2 to l_2;

```
        }
      find all corresponding line segments in the presumed congruent shape
        {
            start from the second intersection to the last intersection of the selected
                shape;
            compare the lengths of corresponding edges (must be equal);
            compare the angle at corresponding vertices (must be equal);
        }
    }
generate a representation of a design composition
    {
      if (there are congruent shapes)
      then (get all the congruent shapes and the initial vertices)
      else (get the selected shape and the initial vertices)
    }
```

These processes are repeated when the designer selects another shape or changes the frame around the design composition. The results of using the Generator module are multiple representations in the form of bounded shapes between the initial boundary and either the recognised congruent shapes or the selected shape produced in DXF files.

5.2 Shape Semantics Recognition using the Recogniser Module

The Recogniser module uses a computable structural shape pattern representation that focuses on shape relationships as well as physical properties. The Recogniser module provides the capability to recognise shape semantics from the multiple representations produced using the Generator module. The Recogniser module detects the shape semantics at each representation and produces an observation for each corresponding representation. The graphical representations of shape semantics recognition are illustrated in Section 4.2.1. The Recogniser detects each representation according to predefined conditions of shape semantics. A shape semantic is found by the Recogniser when all of its preconditions are satisfied. The recognition of shape semantics is performed as follows:

```
compare properties of all shapes within each representation:
shape congruency identification
    {
      compare the number of vertices;
      compare the length of corresponding edges;
      compare the angle at corresponding vertices;
      compare the ratio of each two consecutive edges;
    }
congruent shapes relationships identification
    {
```

draw line segments connecting the corresponding vertices between each two congruent shapes (G_c);
get the slope of each G_c;
generate the midpoint of each G_c;
draw a line segment perpendicular on each G_c from its midpoints G_p;
get the slope of each G_p;
draw a line connecting the midpoints of all G_c G_m;
}

identification of reflective symmetry (M_r) around an axis G_m
{
two shapes must be congruent;
the slopes of all G_c must be equal;
the slopes of all G_p must be equal;
the slope of the G_m must be equal to all G_p;
G_m must occupy the same position as all G_p;
}

identification of reflective symmetry (M_t) around multiple axes
{
at least four shapes must be congruent;
each shape must be reflected around more than one axis;
reflection axes must be perpendicular;
}

identification of simple rotation (R_s)
{
two shapes must be congruent;
the slopes of all G_c are not equal;
the slopes of all G_p are not equal;
the slope of the G_m is not equal to all G_p;
all G_p intersect in a single point (rotation centre);
}

identification of cyclic rotation (R_n)
{
at least three shapes must be congruent;
find a simple rotation between each of two consecutive congruent shapes;
get the rotation angle between each of two consecutive congruent shapes;
congruent shapes are rotated around a single point (rotation centre);
all rotational angles must be equal and none of which should be more than 120^o;
}

identification of translational repetition (P_r)
{
at least three shapes must be congruent;
the slopes of all G_c must be equal;
get the centre of each congruent shape;
get the distance between the centres of each of two consecutive congruent shapes (translational distance);
all the translational distances must be equal;
}

identification of scaling (E_s)
{
the number of vertices of the two shapes must be equal;
the angles at corresponding vertices must be equal;
the ratio of the length between corresponding edges must be equal (scale ratio);
}
identification of adjacency (A_d)
{
get line segments in each polyline shape in the representation with their coordinates;
find line segments that are used more than once;
adjacent shapes must at least share one line segment on their boundaries;
}
identification of dominance (D_m)
{
get the areas for each polyline shape in the representation;
get the total area of the shapes in the design composition;
get the centre of the design composition;
the dominant shape is the largest shape in area within the design composition but at least has to be 1/3 of the total area;
the centre of the design composition must occur within the boundary of the dominant shape;
}
identification of centrality (C_e)
{
get the centre of the design composition;
get the congruent shapes in the design composition;
the centre of the design composition must occur within the boundary of the central shape;
the distances from the centres of the congruent shapes to the centre of the central shape must be equal;
}
identification of radiality (T_r)
{
at least three shapes must be congruent;
find a simple rotation between each of two consecutive congruent shapes;
get the rotation angle between each of two consecutive congruent shapes;
congruent shapes must be rotated around a single point (rotation centre);
rotational angles are not equal;
}
identification of linearity (L_s)
{
at least three shapes must be congruent or scaled shapes;
the slopes of all G_c must be equal;
get the centre of each congruent shape;
get the distance between the centres of each of two consecutive congruent shapes (translational distance);

translational distances are not equal;
}

All of the above preconditions that are necessary and sufficient for recognising shape semantics are tested against each representation. The group of shape semantics found by the Recogniser is considered as a single observation from the corresponding representation. A set of observations is constructed from the corresponding set of multiple representations developed using the Generator module. The Recogniser module is triggered to recognise shape semantics whenever the Generator module generates a representation. The Recogniser module is developed in Windows 95 environment on a PC. It is written in the C++ language and used AutoCAD DXF files generated by the Generator module as its input. Some of its sub-modules were adapted from SPARS (Cha, 1998). The result of using the Recogniser is a set of observations where shape semantics were recognised within the design environment of SLiDe.

5.4 Locating Shape Semantics in relation to their Situations using the Incremental Situator Module

The Situator module locates the recognised shape semantics in relation to their situations by finding the regularities of relationships among them across the observations. These regularities are clustered in terms of situational categories that reflect the relationships and dependencies among shape semantics based upon where they were recognised. Within the situated view of designing, relationships and dependencies change over time whenever changes take place in the environment and concept drift occurs. Concept drift implies that the clustering changes during the period of learning. The conditional probabilities reflected by the new observations change also and are no longer accurately represented by the categories in context-free machine learning systems. The Restructuring Situator updates the previously learned situational categories and restructures the hierarchical tree structure accordingly. So the tasks to be handled by both the Situator and Restructuring Situator modules are defined as follows:

Initial input:

Given:
- A set of observations constructed from multiple representations where each observation is comprised of a group of shape semantics recognised from a single representation.

Find:
- The regularities of relationships among shape semantics and cluster them into situational categories based upon where they were recognised across the observations;
- The summary description of each category that describes its observations; and
- The hierarchical tree structure for those situational categories.

Further Inputs:

Given:
- A new set of observations produced from other representations where each observation is comprised of a group of shape semantics recognised from a single representation.

Find:
- The new regularities from the new observations and their effects on the previously learned situational categories and either accommodate them within the learned categories and update their probabilities or create new situational categories;
- The summary description of each category that describes its observations; and
- The update in the hierarchical tree structure of situational categories considering the new changes in the design environment.

To handle this task an incremental clustering mechanism that uses a nominal description of knowledge is needed. The roles of this incremental clustering mechanism are to categorise shape semantics within which they were recognised across the observations; provide a description for each category; and organise these categories in a hierarchical tree structure. This incremental clustering mechanism should not be affected by the order of observations because it deals with a design environment that changes over time. This means that the learned categories are to be modified in response to the changes in the environment. These conditions form the criteria of the task at hand. In this task neither what is learned nor the environment that SLiDe learns from are fixed. Both change over time as a reflection of the concept of situatedness.

The Situator and Restructuring Situator are integrated as implemented in the Incremental Situator module within SLiDe using the modified unsupervised incremental clustering mechanism in COBBIT (Kilander and Jansson, 1993). The Incremental Situator module is written in the C language and implemented in a Unix environment. 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 and especially the unsupervised incremental clustering mechanism are presented in the following subsections.

5.4.1 An overview of clustering mechanisms

Clustering mechanisms automatically discover categories of observations that are similar in one or more dimensions. Once uncovered, these categories might suggest features that characterise observed knowledge. Ideally, clustering organises a set of observations in a way that facilitates more efficient problem solving. It is not intended here to compare different clustering mechanisms because they 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 for 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. 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, 1996) 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.

5.4.2 Features of the Incremental Situator

The Incremental Situator module in SLiDe has been implemented using the unsupervised incremental clustering mechanism as a utilisation of COBBIT (Kilander and Jansson, 1993). This clustering mechanism uses a nominal description of knowledge, categorises the representations probabilistically, provides a description for each category, organises these categories hierarchically and most importantly is not affected by the order of representations as it deals with environments that change over time. This means the learned categories are modified in response to the change in the environment. There are some distinctive features that identify this unsupervised incremental clustering mechanism. These are introduced in the following subsections.

5.4.2.1 Unsupervised incremental learning

In non-incremental learning methods, all observations must be present at the outset of system execution. In contrast, incremental learning methods accept a stream of observations that are assimilated one at a time. A primary motivation for using incremental learning is that knowledge may be rapidly updated with new observations. Incremental learning methods do not extensively reprocess previously encountered observations while incorporating a new one. Without this constraint, one could make any non-incremental system incremental simply by adding the new observations to an existing set and reapplying the non-incremental method to the extended set.

In the Incremental Situator, the incremental unsupervised learning, as developed in COBWEB, uses an incremental hill-climbing learner. Hill-climbing is a classic AI search method in which one applies all operator instantiations, compares the resulting states using an evaluation function, selects the best state and iterates until no more progress can be made. 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 learner 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 change as it gets new observations. 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.

5.4.2.2 Clustering and learning

Classification and learning are intertwined, as originally developed in COBWEB, with each observation being stored down through a hierarchy and altering this hierarchy in its passage. COBWEB uses a slightly generalised version of Gluck and Corter's (1985) function to control its classification and learning behaviour. Category utility is a heuristic measure to guide the search as a means of predicting the basic level in human classification hierarchies. It favours clustering that maximises the potential of inferring information. In doing this, it attempts to maximise intra-class similarity, predictability, and inter-class differences and predictiveness. It also provides a principled 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. Combining these measures of individual attributes and values into an overall measure of clustering quality specifically as show in expression (1), represents a tradeoff between predictability $P(A_i = V_{ij}|C_k)$ and predictiveness $P(C_k|A_i = V_{ij})$ that has been summed across all classes (k), attributes (i), and values (j).

$$\sum_k \sum_i \sum_j P(A_i = V_{ij})\, P(C_k \mid A_i = V_{ij})\, P(A_i = V_{ij} \mid C_k) \qquad (1)$$

The probability $P(A_i = V_{ij})$ weights the individual values, so that frequently occurring values play a more important role than those occurring less frequently. Using Bayes' rule, we have $P(A_i = V_{ij})\, P(C_k|A_i = V_{ij}) = P(C_k)\, P(A_i = V_{ij}|C_k)$, enabling us to transform expression (1) into an alternative form shown in expression (2).

$$\sum_k P(C_k) \sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 \qquad (2)$$

Gluck and Corter (1985) have shown that the sub-expression $\sum_i \sum_j P(A_i = V_{ij}|C_k)^2$ is the expected number of attribute values that one can correctly guess for an arbitrary member of class C_k. This expectation assumes a probability matching strategy, in which one guesses an attribute value with a probability equal to its probability of occurring. Thus, it assumes that one guesses a value with probability $P(A_i = V_{ij}|C_k)$ and that this guess is correct with the same probability. Building on expression (2), 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 latter term is simply $(\sum_i \sum_j P(A_i = V_{ij})^2)$, so one must subtract this from expression (2). The complete form for category utility is as shown in expression (3):

$$\frac{\sum_{k=1}^{n} P(C_k) \sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2}{n} \qquad (3)$$

The difference between two expected numbers is divided by *n*, the number of categories. This division enables one to compare different size clustering, which must occur whenever considering merging, splitting or creating a new category. The choice of category utility as a heuristic measure dictates a category description other than the logical, typically conjunctive, descriptions used in AI. Category utility can be computed from $P(C_k)$ of each category in partition and $P(A_i = V_{ij}|C_k)$ for each attribute value. A summary description that lists attribute values and associated probabilities is a probabilistic category.

5.4.2.3 Hierarchical tree structure of situational categories

Learned categories are organised and structured in a hierarchy. This type of data structure contains a set of nodes partially ordered by generality. Each node in the hierarchy represents a category and also contains a description of that category. In contrast, most learning from examples (Mitchell, 1982; Michalski and Stepp, 1983) focuses on learning one or a few categories at a single level of abstraction. Methods of constructing decision trees (Quinlan, 1986) are closer in spirit, but lack any explicit descriptions of the nodes themselves. The presence of hierarchical organisation of knowledge suggests an approach for classifying new observations from the environment. One simply begins at the most general (top) node and sorts the new observations down through the hierarchy. This classification method is very similar to that used by the decision tree systems. 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 unsuccessful learning path. The four operators that COBWEB (Gennari et al., 1989; Iba and Langley, 1999) invokes to alter the structure of the clustering's hierarchy are: extending the hierarchy downward; creating disjunct at an existing level; merging two existing classes; and splitting an existing category. This is performed as in the algorithm shown in Table 5.1 and illustrated in Figure 5.3.

5.4.2.4 Manipulation of concept drift

The incremental clustering mechanism in 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, which means that the intention of clustering is not stable during the period of learning. Dependencies of relationships among shape semantics change over time whenever changes take place in the design environment. The conditional probability reflected by the new observations change also and are no longer accurately represented by the categories in the machine learning system. A shared behaviour of systems that track categories intentionally is that they maintain generalisations under a notion of recency and non-monotonicity. The learned situational categories are meant to represent

the situations within which design knowledge (shape semantics), were recognised. Hence, some aspects of the situations might be revealed at different points of time in the observations and usually not in order of their importance. For such situational categories to be appropriately constructed, the learner must integrate these kinds of knowledge to construct the situations without being affected by the order of observations. In order to advance the incremental clustering mechanism in COBWEB to achieve these objectives we selected COBBIT that deals with an environment that changes over time and its clustering structures are not dependent on the order of the observations.

Table 5.1 The algorithm used to alter the structure of the clustering's hierarchy (Gennari et al., 1989).

Input	• The current node *N* of the category hierarchy • An unclassified (attribute-value) Observation (*O*)
Top-level call	• COBWEB (Top-node, *O*)
Variables	• *C*, *P*, *Q* and *R* are nodes in the hierarchy • *U*, *V*, *W* and *X* are clustering scores
	COBWEB(*N*, *O*) **If** N is a terminal node. **Then** Create-new-terminals (*N*, *O*) Incorporate (*N*, *O*) **Else** Incorporate (*N*, *O*) **For** each child *C* of node *N*, Compute the score for placing *O* in *C* Let *P* be the node with the highest score *W* Let *R* be the node with the second highest score Let *X* be the score for placing *O* in a new node *Q* Let *Y* be the score for merging *P* and *R* into one node Let *Z* be the score for splitting *P* into its children **If** *W* is the best score **Then** COBWEB (*P*, *O*) (Place *O* in Category *P*) **Else if** *X* is the best score **Then** initialise *Q*'s probabilities using *O*'s values (place *O* by itself in the new category *Q*) **Else if** *Y* is the best score **Then** let *G* be Merge (*P*, *R*, *N*) COBWEB (*G*, *O*) **Else if** *Z* is the best score **Then** Split (*P*, *N*) COBWEB (*N*, *O*)
Results	• A category hierarchy that classifies the observations

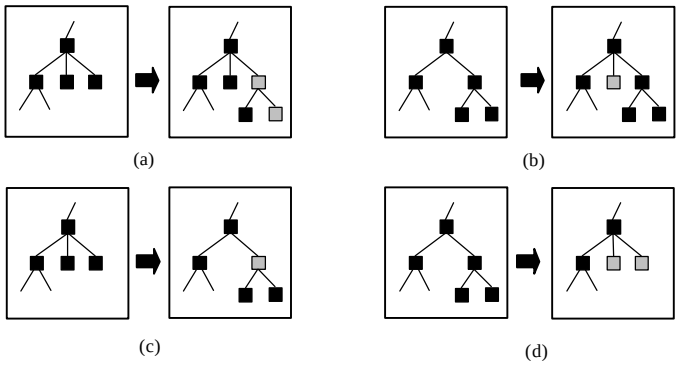


Figure 5.3 Learning operators used to modify the structure of a hierarchy of probabilistic clustering: (a) extending the hierarchy downward; (b) creating a disjunct at an existing level; (c) merging two existing classes; and (d) splitting an existing category. Newly created nodes are shown in grey (Iba and Langley, 1999).

These modifications 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) and performed as shown in Table 5.2. The control parameters of the upper and lower bounds (u and l) 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 a 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. The learned situational categories are continuously influenced by recent observations to confirm (reinforce) or degrade (decay) learned categories or create new ones.

The Restructuring Situator is triggered to update the learned situational categories whenever the Recogniser module detects any new observations from the design environment. The Restructuring Situator facilitates the incremental clustering mechanism in COBBIT to update what has been learned. For instance, if a new set of observations is experienced in SLiDe, the Restructuring Situator analyses the learned knowledge taking into account the new observations and tries either to fit them within the existing categories or to create new categories or sub categories to accommodate them. The restructuring of situational categories is not a mere adding of a new situational category to the existing ones but rather it is an overall restructuring that has included the additional observations and accommodated them in the form of subcategories within the previously learned situational categories. This approach fits well to play the role of the Incremental Situator in SLiDe in which neither what is learned nor the environment that

SLiDe interacts with are fixed. Both change over time as a reflection of the concept of situatedness.

Table 5.2 An extended control algorithm (read-evaluate-learn-trim) using the queue of observations (Kilander and Jansson, 1993).

Begin	• *Object* = the next observation (O) from the input
Loop	• **If** *Object is a training observation* **Then** *Error = Prediction_Error(object)* *root = COBWEB (object, root)* *Queue = Queue + Object* *MaxQSize = ((1-error) * (u-l)) + l* **If** *Length (Queue) > MaxQSize* **Then** $N = 1 + \frac{(length\ (Queue) - MaxQSize)}{4}$ **While** *N > 0* **Do** Extract_Object (Head (*Queue*)) *Queue = Queue* - Head(*Queue*) *N = N* - 1 **Else** Predict missing values and report

5.4.3 Illustration of how the Incremental Situator works

This section presents a simple illustration of how the Incremental Situator constructs the situational categories, its hierarchical tree structure and modifies the hierarchy in response to a new observation. An example will be considered in which a set of observations that consists of four observations constructed by the Recogniser module from a set of four representations generated using the Generator module with the interaction of the designer. The Situator module initialises its hierarchy into a single node, basing the values of this category's attributes on the first observation. Upon encountering a second observation, the Situator averages its values into those of the category and creates two children, one based on the first and another based on the second. The Situator module continues to sort each observation through its clustering hierarchy in which observations are stored as leaves of the clustering hierarchical tree structure and the root node summarises all observations seen in the domain as shown in Figure 5.4. When a new observation is experienced by SLiDe, the Restructuring Situator is triggered to learn commencing with retrieving potential categories. To do so, at each node the Restructuring Situator retrieves all children and considers placing the observation in each of these categories. Each of these constitutes an alternative clustering that incorporates the new observation. Using an evaluation function, it then selects the best clustering. The Restructuring Situator also considers creating a new category that contains only the new observation, and compares this clustering to the best clustering that uses only existing categories. If the clustering based on existing classes wins the competition, the Restructuring Situator modifies the probability of the selected

category and the conditional probabilities of its attribute values. Thus, predictability scores for values occurring in the observation will increase, whereas those for values not occurring will decrease as shown in Figure 5.5. The Restructuring Situator continues to sort the observations down through the hierarchical tree structure, recursively considering the children of the selected category. If the clustering with the singleton class emerges as the winner, the Restructuring Situator creates this new category and makes it a child of the current parent node. The system bases the values for this new category's attributes on those found in the observation, giving them each predictability scores of one as shown in Figure 5.6. The results of using the Incremental Situator module are a hierarchical tree structure of the learned category along with a description of the regularity for each situational category generated and drawn in a postscript file.

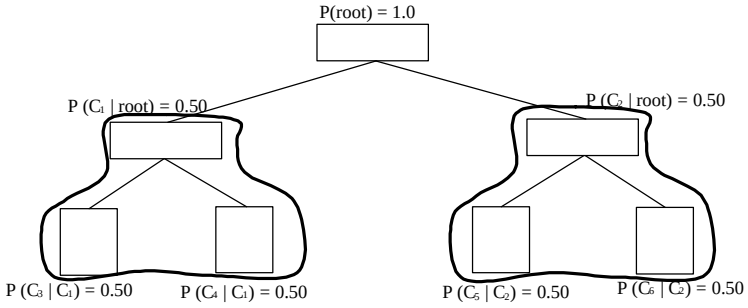


Figure 5.4 A hierarchical tree constructed by the Situator module.

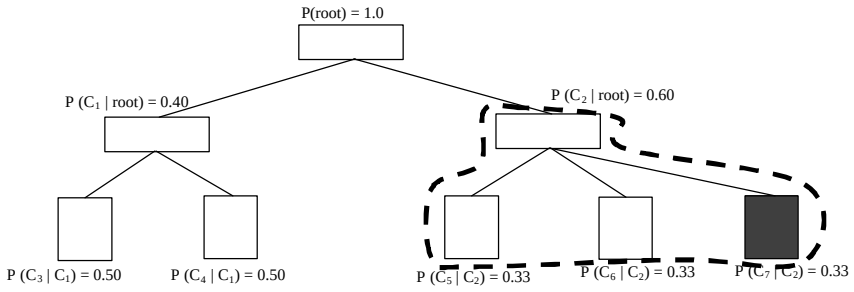


Figure 5.5 A revised hierarchical tree constructed by the Restructuring Situator module.

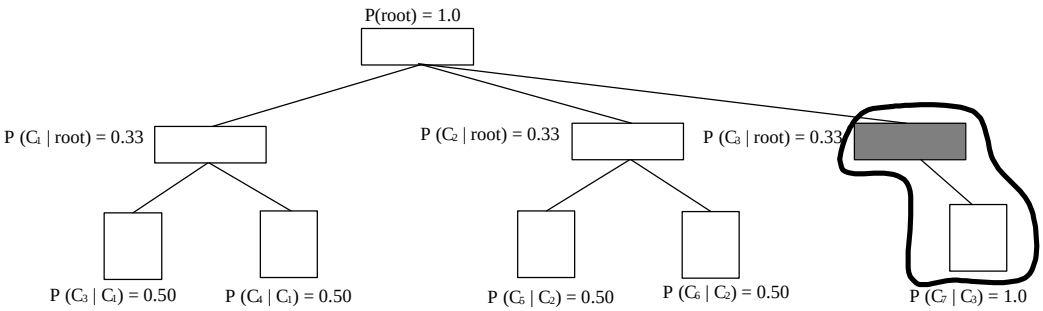


Figure 5.6 A revised hierarchical tree constructed by the Restructuring Situator module.

Chapter 6

Application of SLiDe

This chapter presents a demonstration of using SLiDe to learn the applicability conditions, ie situatedness, of shape semantics within which they were recognised from sets of observations constructed from various representations. SLiDe modules are used to generate multiple representations of a design composition, detect and recognise the shape semantics in these representations, construct a set of observations from the corresponding representations and learn the applicability conditions of these shape semantics in the form of situational categories. These situational categories represent the regularities of relationships among shape semantics across the observations. SLiDe updates what it has learned in response to the newly constructed observations. The update is in terms of reinforcing, decaying or reconstructing what has been previously learned.

6.1 Introduction

The demonstration of SLiDe in this Chapter aims at showing its capabilities to generate multiple representations of a design composition; recognise shape semantics from these representations; construct sets of observations from the corresponding representations; and learn the situatedness of shapes semantics. The regularities of these relationships are clustered in the form of situational categories. The situatedness of shape semantics is updated in response to the new observations of the design composition. The demonstration of SLiDe proceeds in the following sequence:

(i) Select an example for an architectural design composition and draw an initial representation of it;
(ii) Use the Generator module to develop a set of representations from the initial representation of the selected design composition;
(iii) Use the Recogniser module to detect shape semantics in each representation and construct a set of observations for the corresponding representations accordingly;
(iv) Use the Incremental Situator module to learn the situatedness of shape semantics through learning the regularities of relationships among shape semantics across the observations and cluster them in the form of situational categories;
(v) Use the Generator module to produce another set of representations of the same design composition;
(vi) Use the Recogniser module to construct a second set of observations;

(vii) Use the Incremental Situator module to update the situatedness of recognised shape semantics and accommodate the new observations; and

(viii) Repeat the steps (v), (vi) and (vii) three or more times and monitor the updates of the learned situational categories (situatedness of shape semantics), and the overall restructuring of their hierarchical tree structure.

6.2 Selection of an Architectural Design Composition

SLiDe manipulates architectural design compositions composed of linearly bounded shapes. An architectural design composition of the Exter Library (in the form of a floor plan), designed by Louis Khan in New Hampshire as shown in Figure 6.1 is selected as an example to demonstrate the capabilities of SLiDe.

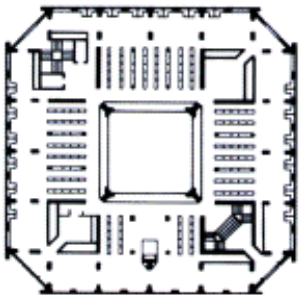
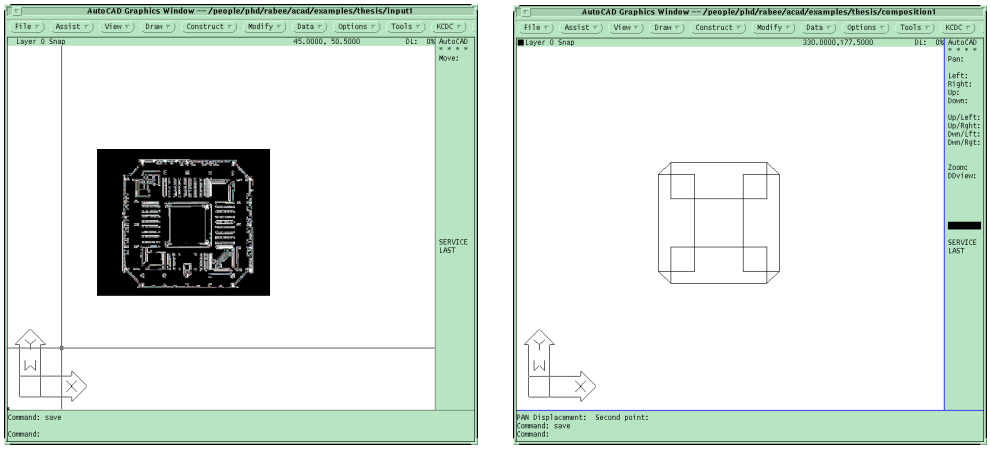


Figure 6.1 An example of architectural design composition: The Exter Library, in the form of a floor plan, designed by Louis Khan in New Hampshire (Clark and Pause, 1996).

6.3 Development of Multiple Representations

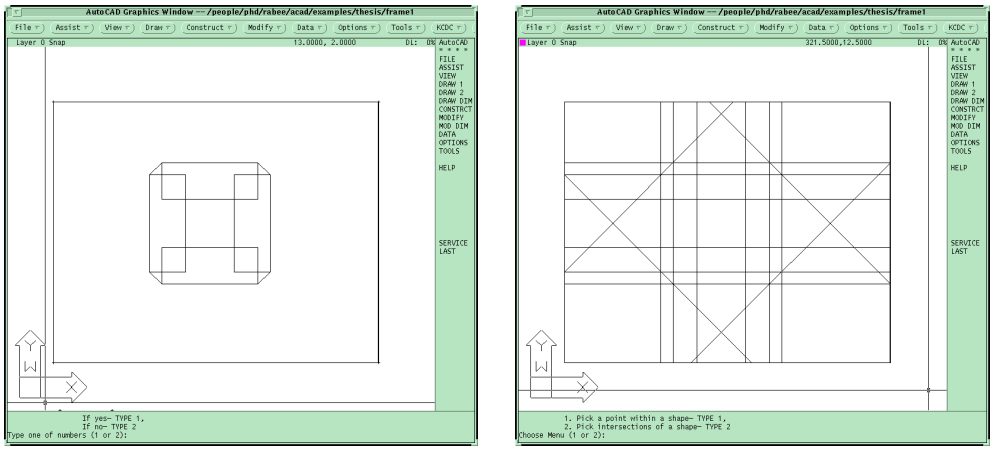
The selected design composition is scanned and imported into AutoCAD as shown in Figure 6.2(a). The imported design composition is simplified by the designer redrawing an initial representation (in the form of line segments) into AutoCAD as shown in Figure 6.2(b). The Generator module is loaded and commences with this initial representation of the design composition. The Generator requests the designer to draw a frame around the design composition. The selected frame is shown in Figure 6.3(a). The Generator re-represents the design composition in the form of infinite maximal lines. The Generator module achieves this by extending the line segments to the boundary of the frame drawn by the designer as shown in Figure 6.3(b). The intersections of the infinite maximal lines within this frame are detected and saved by the Generator module defining the design space to be searched. This module then requests the designer to select a shape of interest from among these intersections of infinite maximal lines. The Generator module searches the design space for any congruent shapes corresponding to the selected shape and highlights all the corresponding congruent shapes in shaded areas so as to be easily perceivable by the designer. The Generator does not allow for overlapping shapes while searching for corresponding congruent shapes. Then, the Generator module produces the first representation (N_1) composed of the corresponding congruent shapes and the leftover of the initial representation. Figures 6.4 (a), (b) and (c) show the dimensional and geometrical constraints of the selected shape, highlighting

the selected shape and the first representation (N_1) developed using the Generator module respectively. The designer continues to select more shapes of interest, one at a time. Then, the Generator module continues to search the design space and generates a set of representations, in the form of bounded polyline shapes, each of which is composed of corresponding congruent shapes of the designer's selected shape and the leftover of the initial representation. The first set of multiple representations is completed when the designer decides not to select any more shapes at that time. The results of using the Generator module to produce other different representations of the design composition are shown in Figures 6.5(a) to 6.5(g) showing the representations from (N_2) to (N_9) respectively.



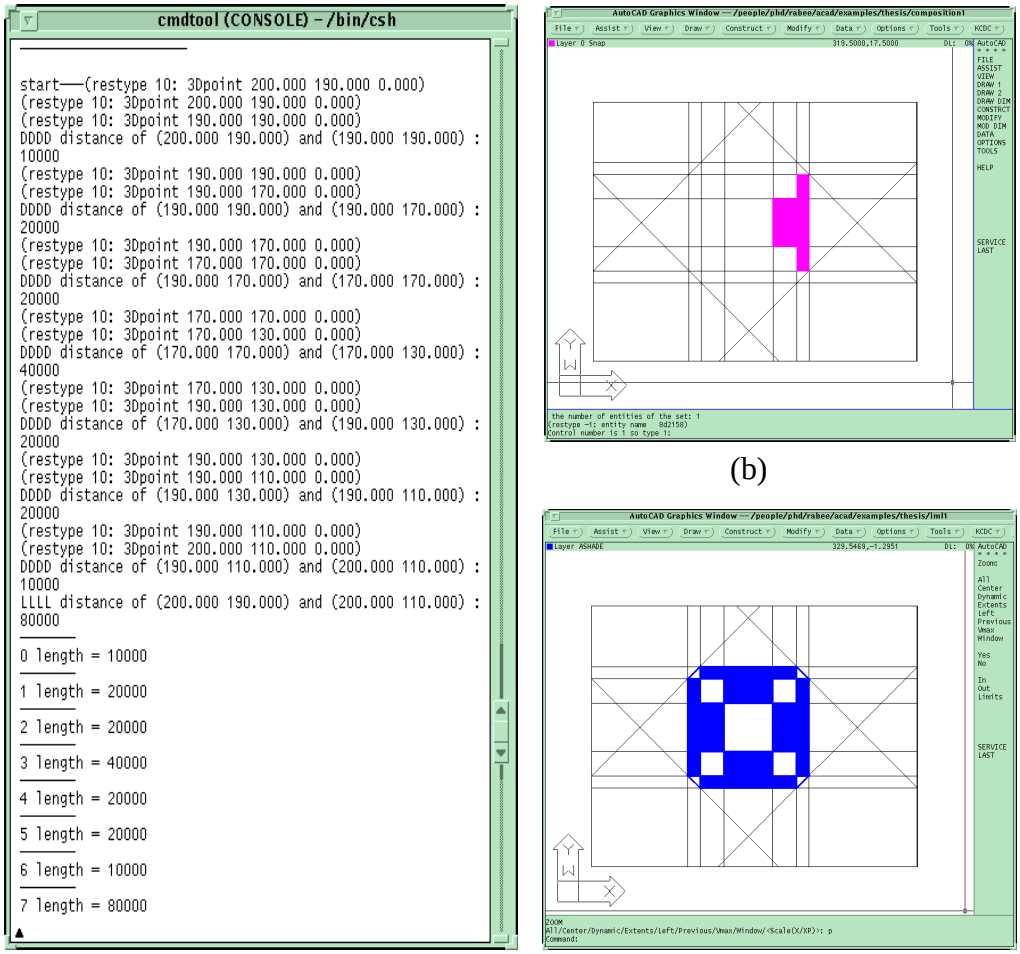
(a) (b)

Figure 6.2 (a) Scanned architectural design composition in Figure 6.1 is imported in AutoCAD, (b) a simplified design composition is drawn by the designer (user) in AutoCAD and serves as an initial representation.



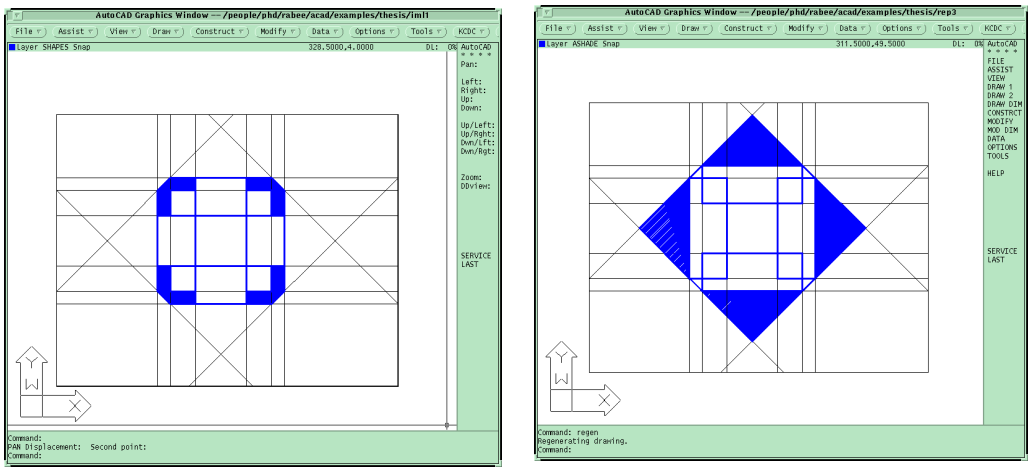
(a) (b)

Figure 6.3 (a) Selected frame drawn by the designer around the design composition, (b) infinite maximal lines of the design composition produced by the Generator module.



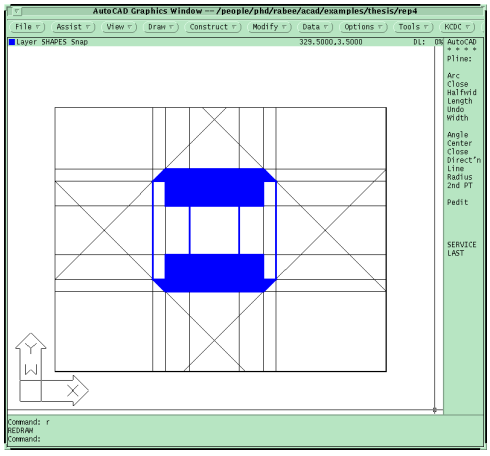
(a) (b) (c): N_1

Figure 6.4 The Generator module produces: (a) dimensional and geometrical constraints of the selected shape, (b) highlighting the selected shape by the designer, and (c) first developed representation (N1).

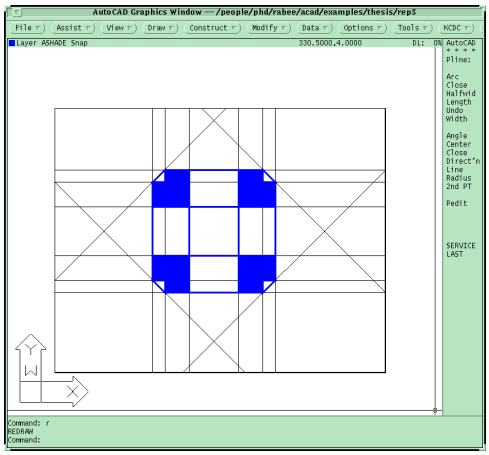


(a): N_2 (b): N_3

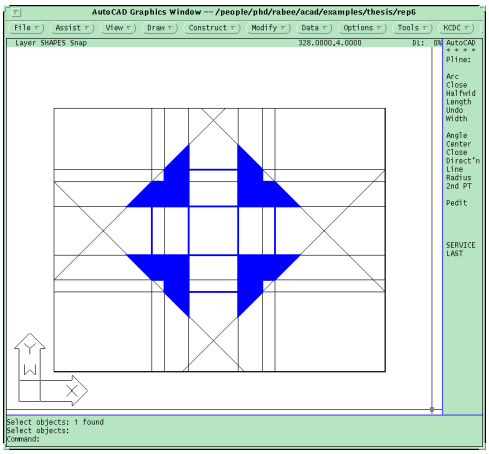
Figure 6.5 (a-b) A set of multiple representations developed using the Generator module through the interaction with the designer to select shapes of interest: (a) and (b) show the representations N_2 and N_3.



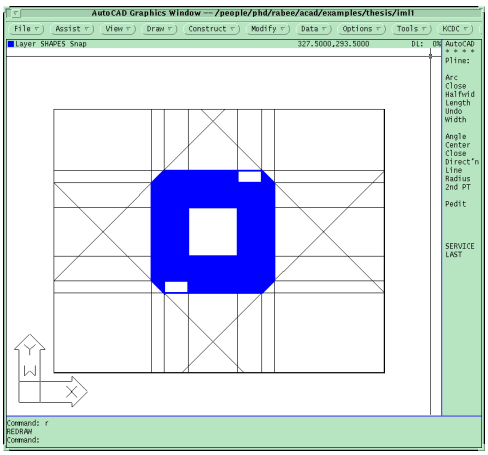
(c): N_4



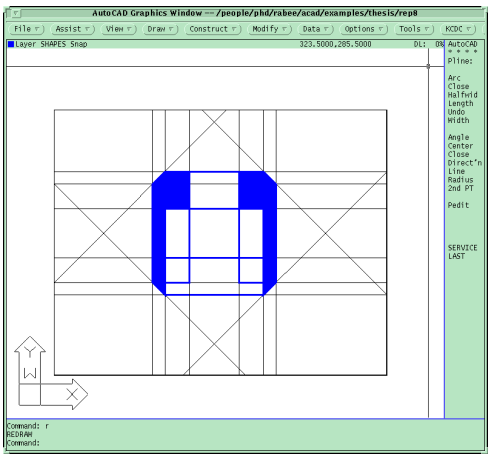
(c): N_5



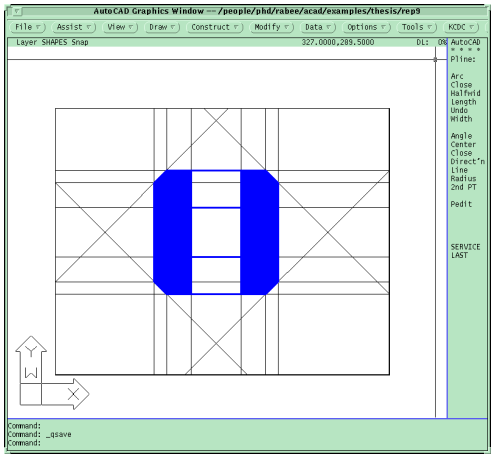
(d): N_6



(e): N_7



(f): N_8



(g): N_9

Figure 6.5 (c-g) A set of multiple representations developed by the Generator module through the interaction with the designer to select shapes of interest: from (c) to (g) show the representations from N_4 to N_9 respectively.

6.4 Constructing a set of Observations from the Developed set of Representations

A set of representations developed using the Generator module forms the initial design environment for the Recogniser module to interact with and learn from. The Recogniser module detects shape semantics at each representation and produces a corresponding observation. Some examples of recognised shape semantics in some of the representations are shown in Figures 6.6 and 6.7. The Recogniser module constructs each observation from the accumulation of recognised shape semantics in each representation. Consequently, after detecting the first set of representations, a corresponding set of observations is constructed. Table 6.1 shows the set of observations (O_n) produced by the Recogniser module from the set of representations developed using the Generator module as shown in Figures 6.4 and 6.5. In Table 6.1, $\boldsymbol{M_r}$, $\boldsymbol{M_t}$, $\boldsymbol{R_s}$, $\boldsymbol{R_n}$, $\boldsymbol{A_d}$, $\boldsymbol{C_e}$ and $\boldsymbol{D_m}$ refer to reflective symmetry around an axis, reflective symmetry around more than one axis, simple rotation, cyclic rotation adjacency, centrality and dominance respectively. The Recogniser module manipulates a wide range of architectural design compositions in linearly bounded shapes. Examples of the Recogniser capabilities to recognise various types of shape semantics from different design compositions are shown in Appendix A.

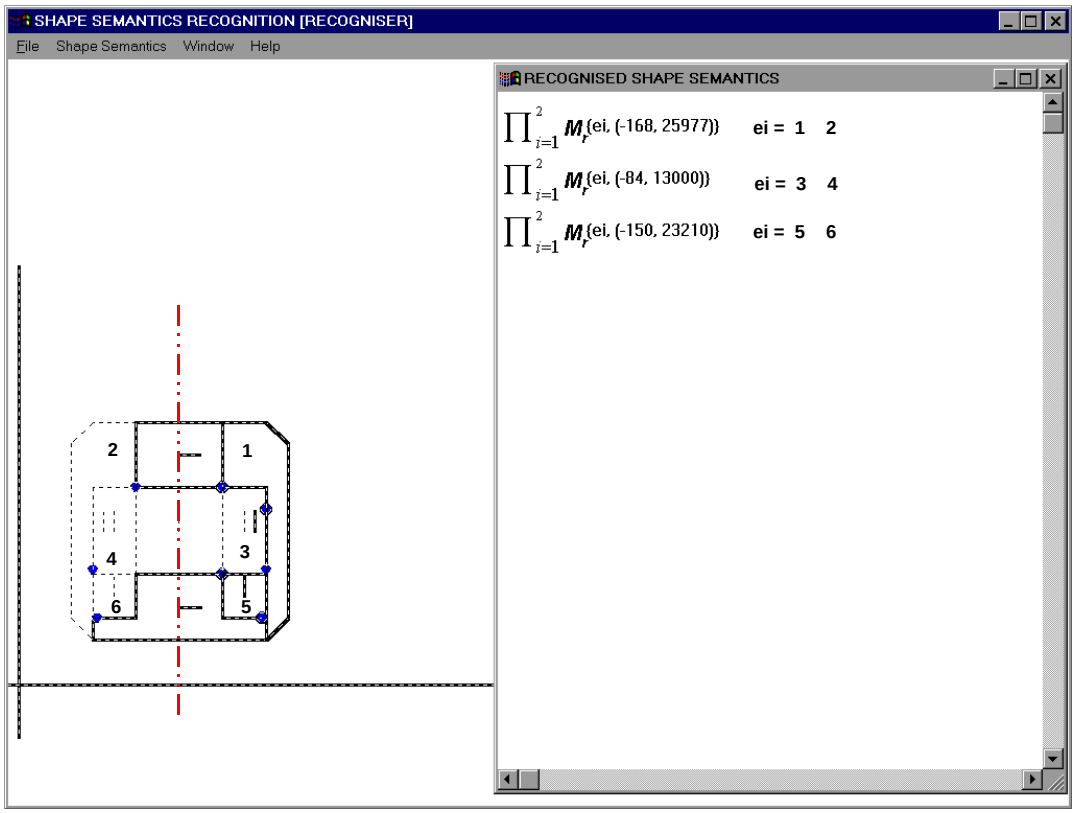
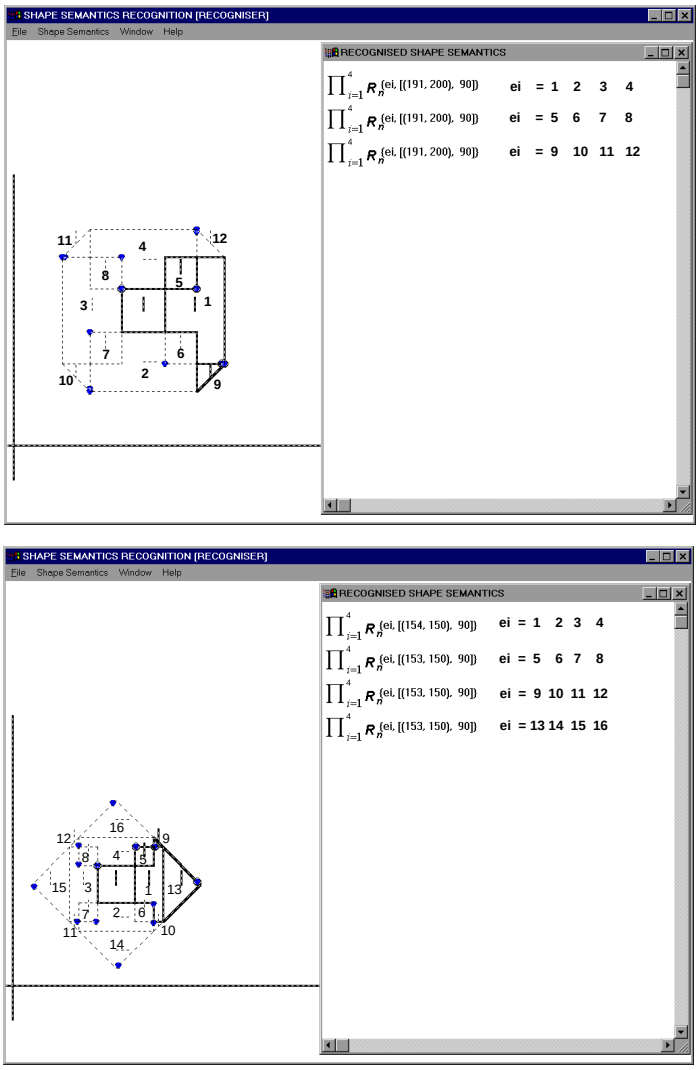


Figure 6.6 An example of reflective symmetry ($\boldsymbol{M_r}$) around an axis found by the Recogniser module in the representation N_8.



(a)

(b)

Figure 6.7 Examples (a) and (b) of cyclic rotation ($\boldsymbol{R_n}$) found by the Recogniser module in the representations N_1 and N_3.

Table 6.1 The first set of observations produced using the Recogniser module to detect shape semantics in the developed representations shown in Figures 6.4 and 6.5.

Representation No.	Corresponding Observation (O_n)	
N_1	O_1	$\boldsymbol{M_t, R_n, C_e, A_d}$
N_2	O_2	$\boldsymbol{M_t, R_n, C_e, A_d}$
N_3	O_3	$\boldsymbol{M_t, R_n, C_e, A_d}$
N_4	O_4	$\boldsymbol{M_r, C_e, A_d}$
N_5	O_5	$\boldsymbol{M_t, R_n, C_e, A_d}$
N_6	O_6	$\boldsymbol{M_t, R_n, C_e, A_d}$
N_7	O_7	$\boldsymbol{R_s, A_d}$
N_8	O_8	$\boldsymbol{M_r, C_e, A_d}$
N_9	O_9	$\boldsymbol{M_r, C_e, A_d}$

6.5 Learning the Situatedness of Recognised Shape Semantics

The set of observations constructed using the Recogniser module is used as the first input to the Incremental Situator module to learn the situatedness of shape semantics in relation to the situations within which they were recognised. The situatedness of shape semantics is learned in the form of the regularities of relationships among shape semantics across the observations. These regularities are clustered in a hierarchical tree structure. The Incremental Situator module employs its incremental unsupervised clustering mechanism to search for the regularities of relationships among shape semantics in the observations and categorises them in relation to situations within which they were recognised. There are no prior categories defined to the Incremental Situator module to measure the match or mismatch of these observations to them but rather they are constructed based on the regularities found in these observations. This reflects that the situatedness of knowledge is not predetermined but rather constructed. The result of using the Incremental Situator module after exposure to the first set of observations is indicated in Figure 6.8. The graph shown in Figure 6.8 is one of the automated forms produced by the Incremental Situator module, in a reduced format, representing the situational categories and their hierarchical tree structure. Each situational category is associated with a summary description of the regularity in its observations.

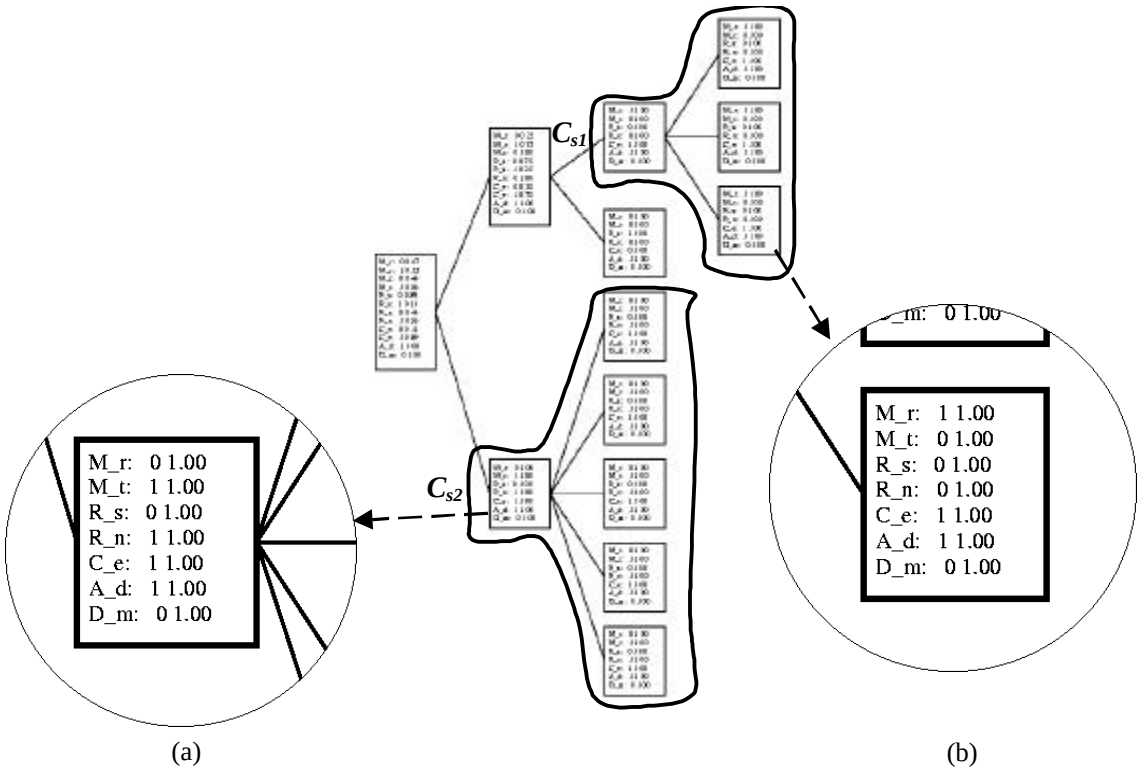


Figure 6.8 Two situational categories C_{s1} and C_{s2} learned by the Incremental Situator module from the set observations constructed by the Recogniser module as shown in Table 6.1; (a) shows a summary description of the regularity of relationships among shape semantics across some observations; (b) shows an observation composed of a group of shape semantics recognised from a representation.

In Figure 6.8, there are two learned situational categories C_{s1} and C_{s2} each of which represents a regularity across some of the observations among a group of shape semantics within which they were recognised. C_{s1} represents the regularity of relationships among the shape semantics M_r, C_e and A_d that refer to reflective symmetry around an axis, centrality and adjacency respectively. C_{s2} represents the regularity of relationships among the shape semantics M_t, R_n, C_e and A_d where M_t refers to reflective symmetry around more than one axis and R_n refers to cyclic rotation. The entities within a situational category, in a state description of a particular state of designing, are candidates for both knowledge and the situation. For instance, in C_{s2} each shape semantic could be the knowledge in focus and all the remaining entities within this situational category become candidates for the situation of that knowledge. The situatedness of design knowledge serves as the applicability conditions of that knowledge within which it was recognised. Thus, the applicability conditions have the potential to guide the use of these shape semantics. For instance, choosing reflective symmetry (M_t) to be the current knowledge in focus F_1 then the other entities of C_{s2} which are R_n, C_e and A_d construct the situation t_1 within which M_t was recognised and serves as the applicability conditions, ie situatedness, of reflective symmetry. In other words, M_t is situated within these shape semantics R_n, C_e and A_d in the design environment. Alternatively, as illustrated in Figure 6.9, if centrality (C_e) is chosen to be the knowledge in focus F_2 then the other entities of C_{s2} which are M_t, R_n and A_d construct the situation t_2 of C_e within which it was recognised. This is a duality between the entities within the same situational category, ie duality between knowledge and the situation. Another example of duality from another learned situational category C_{s1} is shown in Figure 6.10.

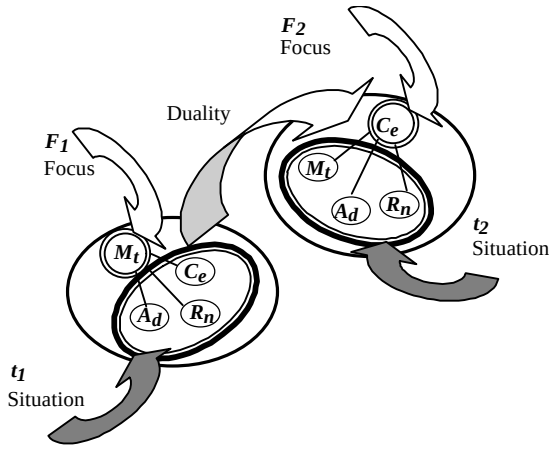


Figure 6.9 An example of the duality between knowledge and situation within the situational category C_{s2}.

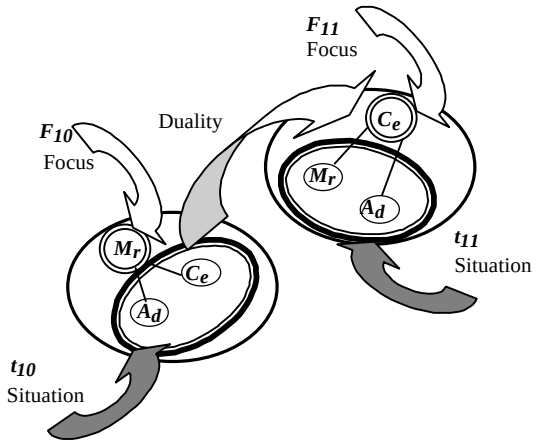
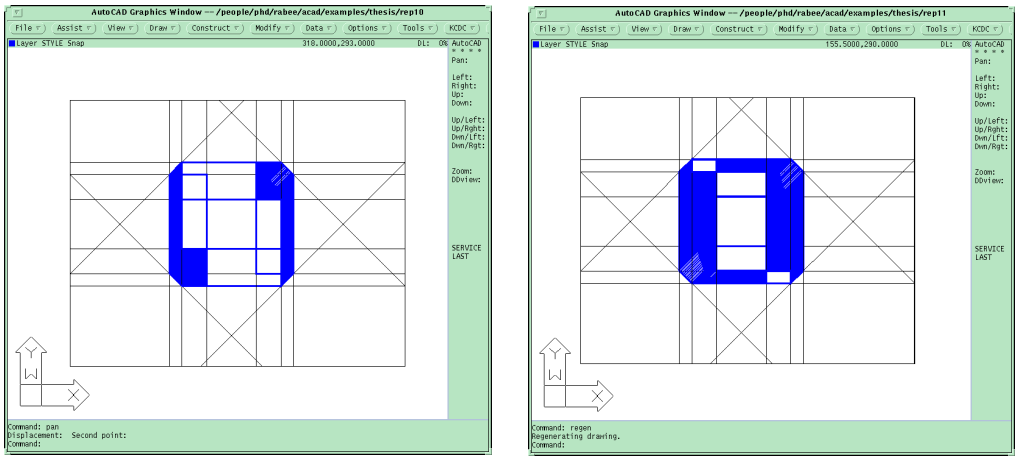


Figure 6.10 An example of the duality between knowledge and situation within the situational category C_{s1}.

A certain piece of knowledge could be recognised in more than just one situation. This means that there are more than one group of applicability conditions or different kinds of situatedness for a single piece of knowledge based on where it was recognised. For instance, from looking at both Figures 6.9 and 6.10, it may be seen that centrality (C_e) has two groups of applicability conditions, ie has two different situations t_2 and t_{11}.

6.6 Incremental Learning about the Situatedness of Shape Semantics

Due to the fluid, dynamic and situated nature of designing, changes take place in the design composition while designing. Such changes have an effect on the relationships among shape semantics and the situatedness of shape semantics are influenced correspondingly. The effect on the situatedness of shape semantics could be in terms of reinforcing, decaying or reconstructing what has been learned previously or new ones could emerge from the new observations. One of the ways in which these changes could be perceived is that the designer is further interested to re-interpret the same design composition which leads to developing other representations of the same design composition. The Generator module is used to satisfy the designer's interest in developing further representations and the result is a new set of representations as shown in Figure 6.11. The generation of this new set of representations triggers the Recogniser module to detect shape semantics in these representations and construct a corresponding new set of observations as shown in Table 6.2.



(a): N_{10} (b): N_{11}

Figure 6.11 A second set of representations developed using the Generator module with the interaction of the designer to select other shapes of interest: (a) and (b) show the representations N10 and N11.

Table 6.2 The second set of observations produced using the Recogniser module to detect shape semantics in the generated representations shown in Figure 6.11.

Representation No.	Corresponding Observation (O_n)	
N_{10}	O_{11}	$\boldsymbol{M_r}$, $\boldsymbol{R_s}$, $\boldsymbol{C_e}$, $\boldsymbol{A_d}$
N_{11}	O_{11}	$\boldsymbol{M_r}$, $\boldsymbol{R_s}$, $\boldsymbol{C_e}$, $\boldsymbol{A_d}$

The Incremental Situator module is triggered to update what has been learned whenever the Recogniser module constructs any new set of observations from the design environment. The Incremental Situator module facilitates the incremental clustering

mechanism to update what has been learned previously taking into account the new observations and trying either to associate them within the existing categories or to create new categories or sub categories to accommodate them. Figure 6.12 shows the learning results produced by the Incremental Situator module after having this new set of observations as a new input. It created a new situational category C_{s3} that clustered both of the two new observations into one new situational category. This is based upon the regularities of relationships among shape semantics found in them and their distinction from the previous observations. In Figure 6.12, there is a new learned situational category C_{s3} accommodating the new set of observations. C_{s3} represents the regularity of the relationships among the shape semantics M_r, R_s, C_e and A_d where R_s refers to simple rotation. An illustration of this regularity and an example of the duality among its entities are show in Figure 6.13 wherein it may be seen that there is another situation within which the centrality (C_e) was recognised.

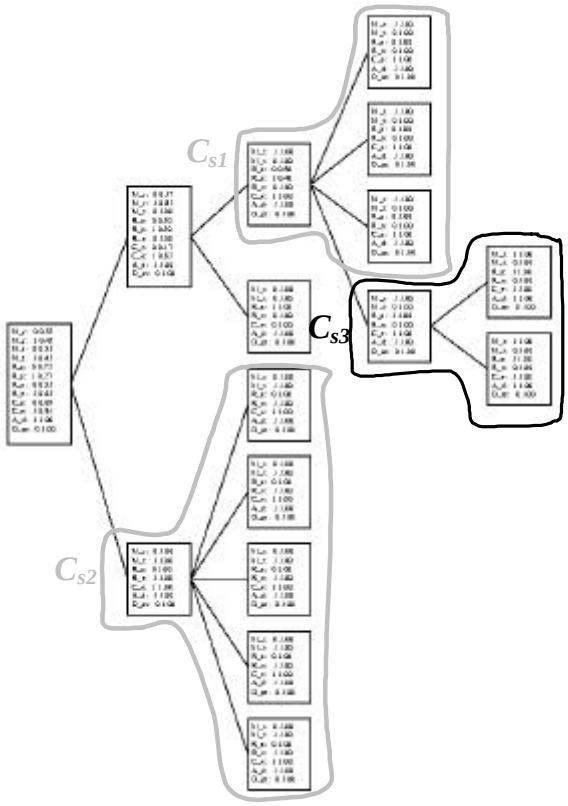


Figure 6.12 A newly learned situational category C_{s3} emerged in response to the second set of observations.

Assuming that once again while designing, the designer has chosen to develop further representations of the same design composition using the Generator module. This constitutes the third set of representations as shown in Figure 6.14. Thus, the Recogniser module is triggered to detect the shape semantics in these representations and constructs a corresponding set of observations as shown in Table 6.3. The Incremental Situator module accommodated the new set of observations into two new categories C_{s4} and C_{s5} as shown, in a reduced format, in Figure 6.15.

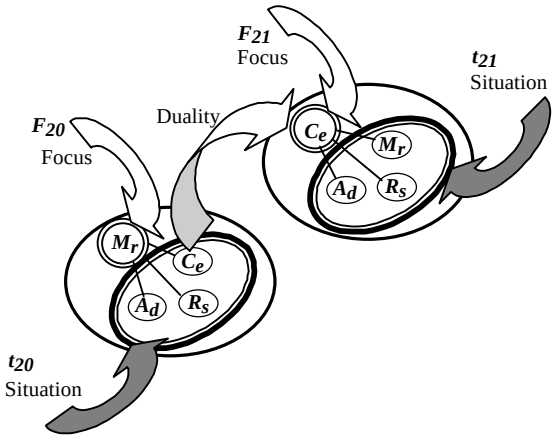
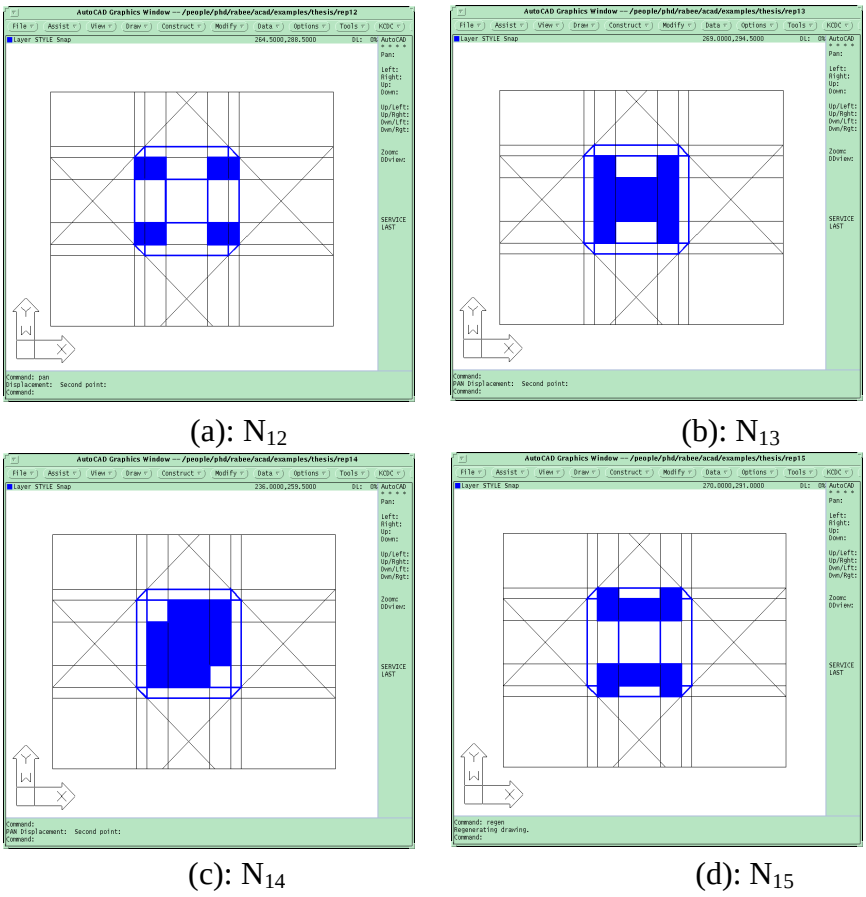


Figure 6.13 Newly learned situational category C_{s3} and an example of the duality among its entities, knowledge in focus and its situation.



(a): N_{12} (b): N_{13}

(c): N_{14} (d): N_{15}

Figure 6.14 The third set of representations developed using the Generator module through the interaction with the designer to select other shapes of interest: (a) to (d) show the representations form N_{12} to N_{15}.

Table 6.3 The third set of observations produced using the Recogniser module to detect shape semantics in the generated representations shown in Figure 6.14.

Representation No.	Corresponding Observation (O_n)	
N_{12}	O_{12}	$\boldsymbol{M_r}$, $\boldsymbol{M_t}$, $\boldsymbol{R_n}$, $\boldsymbol{C_e}$, $\boldsymbol{A_d}$
N_{13}	O_{13}	$\boldsymbol{M_t}$, $\boldsymbol{R_n}$, $\boldsymbol{C_e}$, $\boldsymbol{A_d}$, $\boldsymbol{D_m}$
N_{14}	O_{14}	$\boldsymbol{M_t}$, $\boldsymbol{R_n}$, $\boldsymbol{C_e}$, $\boldsymbol{A_d}$, $\boldsymbol{D_m}$
N_{15}	O_{15}	$\boldsymbol{M_r}$, $\boldsymbol{M_t}$, $\boldsymbol{R_n}$, $\boldsymbol{C_e}$, $\boldsymbol{A_d}$

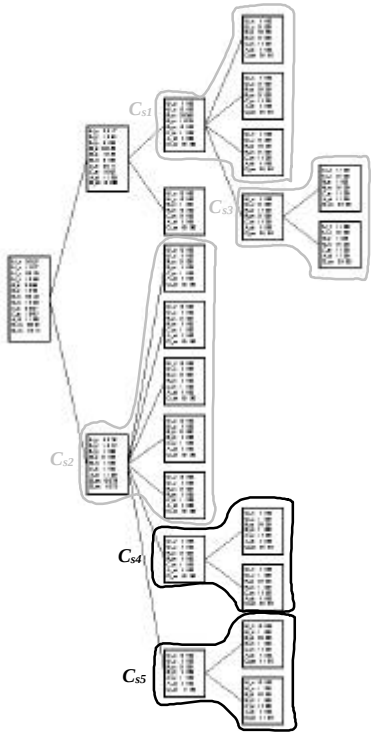


Figure 6.15 Two new learned situational categories $\boldsymbol{C_{s4}}$ and $\boldsymbol{C_{s5}}$ emerged in response to the third set of observations.

The results shown so far from using the Incremental Situator module demonstrate how the new observations were accommodated by adding new situational categories to the existing ones (previously learned), due to their distinctive difference from previous observations. On the other hand, from other sets of observations there is the possibility of reconstructing the situatedness of design knowledge. Since it is not known at the time of learning what is useful knowledge and what is not, all regularities need to be treated as potentially useful knowledge. The Incremental Situator module has the capability to restructure the hierarchy of its situational categories while accommodating the new set of observations. For instance, considering a fourth set of representations developed using the Generator module as shown in Appendix B, the Recogniser module constructed the fourth set of observations accordingly as shown in Table 6.4. The Incremental Situator module accommodated this new set of observations into two new categories $\boldsymbol{C_{s6}}$ and $\boldsymbol{C_{s7}}$ and reinforced what has been learned with the situational

category $*C_{s4}$. This is achieved not only through just adding these two categories to the existing structure of previously learned situational categories but rather restructuring the overall hierarchy to include these new categories as shown in Figure 6.16. Furthermore, from a fifth set of representations produced using the Generator module as shown in Appendix C, the Recogniser module constructed the fifth set of observations shown in Table 6.5. Based on this set of observations the Incremental Situator module reconstructed the situatedness of what had been learned previously, produced a reconstructed situational category $C_{s1(a)}$ and reinforced what had been previously learned in both $*C_{s2}$ and $*C_{s3}$ in addition to creating a new situational category C_{s8} as shown in Figure 6.17.

Table 6.4 The fourth set of observations produced using the Recogniser module to detect shape semantics in the generated representations shown in Appendix B.

Representation No.	Corresponding Observation (O_n)	
N_{16}	O_{16}	M_r , M_t , R_s , R_n, C_e , A_d
N_{17}	O_{17}	M_r , M_t , R_s , R_n, C_e , A_d
N_{18}	O_{18}	M_r , M_t , R_n, C_e , A_d
N_{19}	O_{19}	M_r , M_t , R_n, C_e , A_d
N_{20}	O_{20}	M_r , M_t , R_n, C_e , A_d
N_{21}	O_{21}	M_r , M_t , R_n, C_e , A_d
N_{22}	O_{22}	M_r , M_t , R_n, C_e , A_d
N_{23}	O_{23}	R_n , C_e , A_d
N_{24}	O_{24}	R_n , C_e , A_d

Table 6.5 The fifth set of observations produced using the Recogniser module to detect shape semantics in the generated representations shown in Appendix C.

Representation No.	Corresponding Observation (O_n)	
N_{25}	O_{25}	M_r , R_s, C_e , A_d
N_{26}	O_{26}	R_s, C_e , A_d
N_{27}	O_{27}	M_r , R_s, R_n, C_e , A_d
N_{28}	O_{28}	M_t , R_n, C_e , A_d
N_{29}	O_{29}	M_t , R_n, C_e , A_d
N_{30}	O_{30}	M_t , R_n, C_e , A_d
N_{31}	O_{31}	M_t , R_n, C_e , A_d
N_{32}	O_{32}	M_r , C_e , A_d
N_{33}	O_{33}	M_r , C_e , A_d
N_{34}	O_{34}	M_t , R_n, C_e , A_d
N_{35}	O_{35}	M_t , R_n, C_e , A_d
N_{36}	O_{36}	M_t , R_n, C_e , A_d
N_{37}	O_{37}	M_t , R_n, C_e , A_d
N_{38}	O_{38}	M_t , R_n, C_e , A_d
N_{39}	O_{39}	M_r , M_t , R_n, C_e , A_d

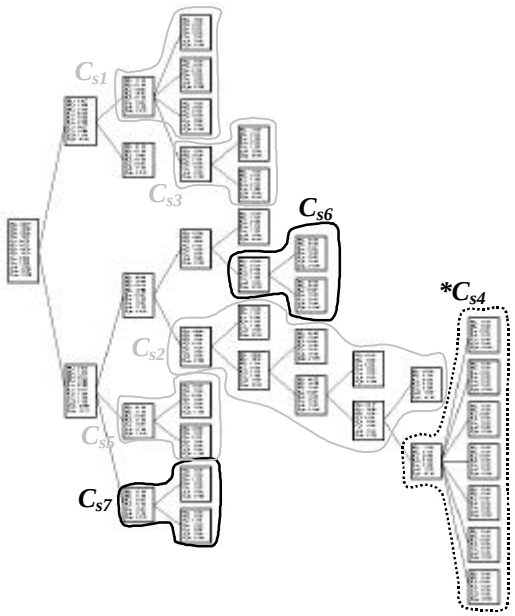


Figure 6.16 An overall restructuring of the hierarchical tree structure wherein situational categories C_{s6} and C_{s7} emerged and $*C_{s4}$ is reinforced (shown dotted), in response to the fourth set of observations shown in Table 6.4.

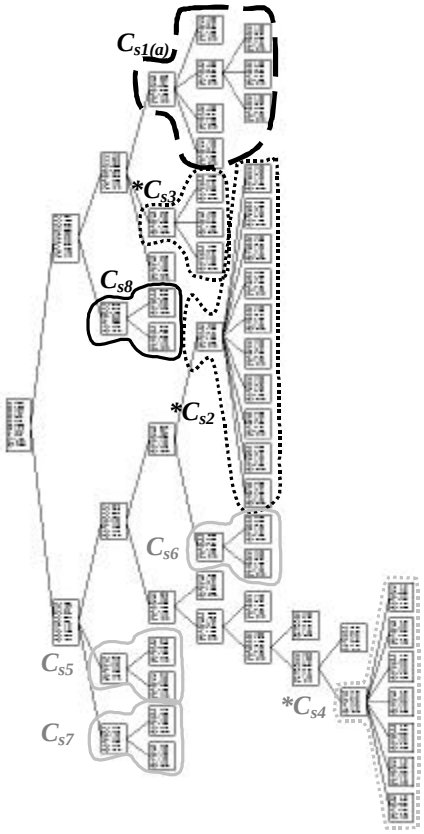


Figure 6.17 Reconstructing a previously learned situational category $C_{s1(a)}$ (shown dashed); reinforcing $*C_{s2}$ and $*C_{s3}$ (shown dotted); and creating a new situational category C_{s8} in response to the most recent (fifth), additional set of observations shown in Table 6.5.

The Incremental Situator module has been implemented with a menu driven interface that allows the user to navigate the learned situational categories presented in the graph (hierarchical tree structure). The interface allows the user to change the orientation of the graph on the screen either vertically or horizontally as shown in Figures 6.18(a) and (b) respectively. Some statistics about the hierarchical tree structure of the learned categories from the first observation are shown in Figure 6.18(c). These statistics cover the created nodes on the hierarchical tree structure and the merge and split of situational categories that occurred during learning. Figure 6.19(a) shows the result of using the Restructuring Situator module to update what had been learned and restructure the hierarchical tree accordingly in response to new sets of observations. Figure 6.19(b) shows some statistics about the updated hierarchical tree structure in which merging occurs between situational categories. The interface allows for zooming in and out to see the description of each of the observations and the categories as shown in Figure 6.20.

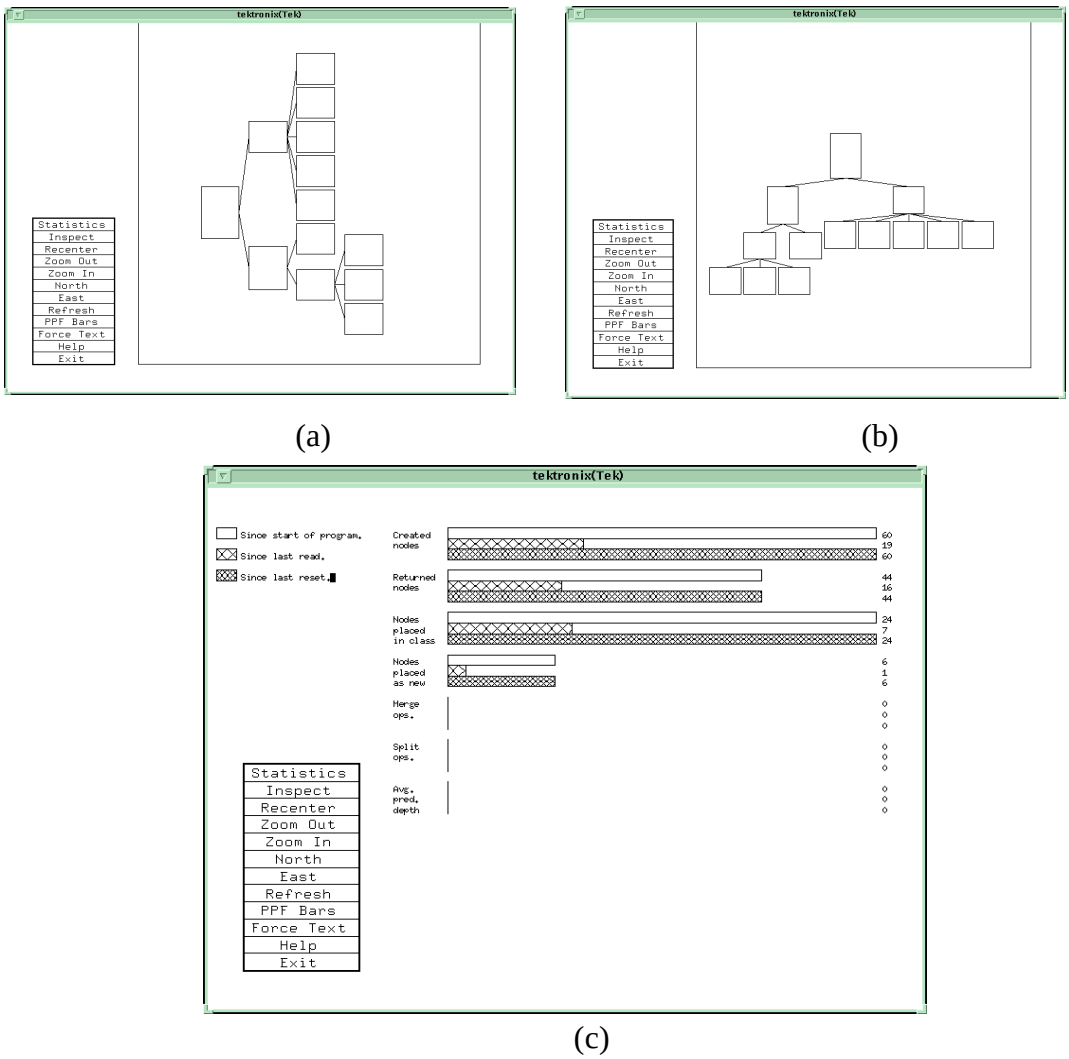
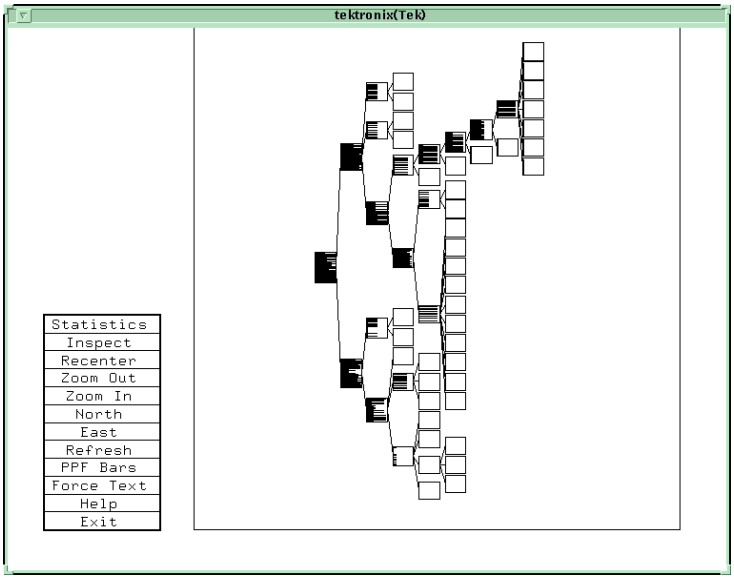
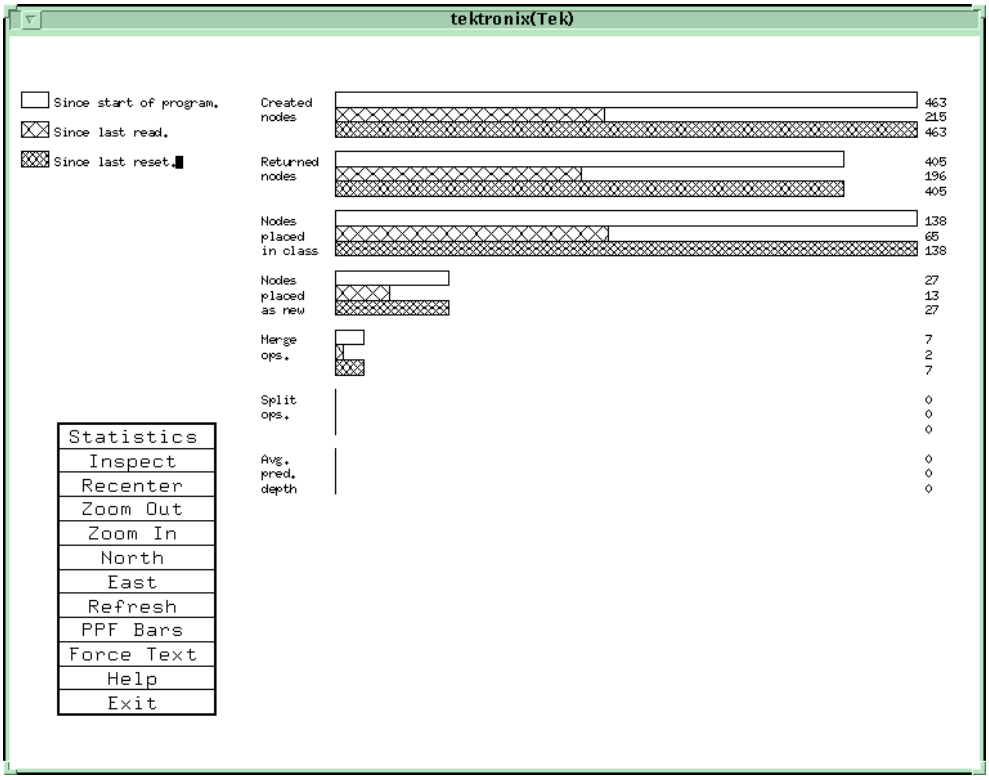


Figure 6.18 (a) and (b) Snapshots of the hierarchical tree structure constructed from the first set of observations using the Situator module shown in a vertical and a horizontal direction respectively and (c) some statistics about this hierarchical tree structure.



(a)



(b)

Figure 6.19 (a) shows the result of updating what had been learned and restructuring the hierarchical tree accordingly in response to the fifth set of observations and (b) shows some statistics about the updated hierarchical tree structure in which merging occurs between situational categories.

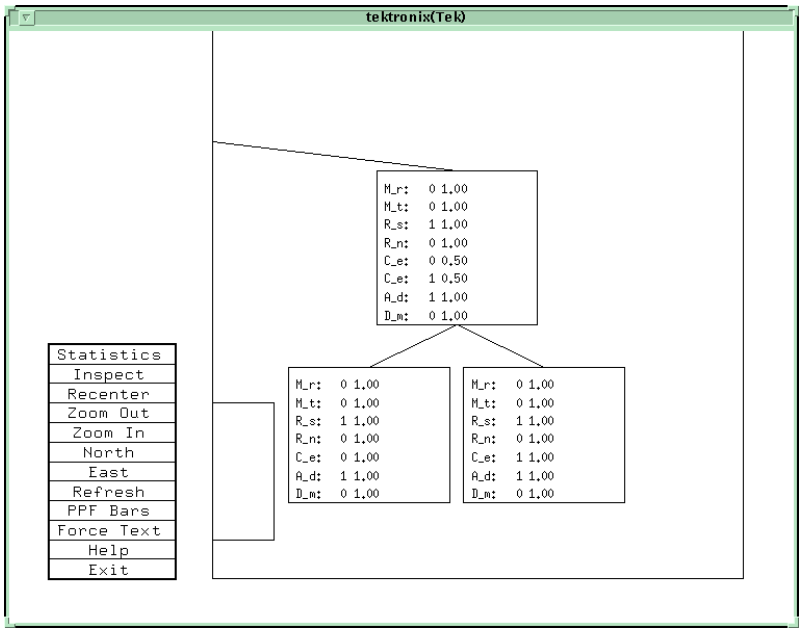


Figure 6.20 Zooming in within the graph to see the description of the observations and their category.

6.7 Discussion

This demonstration illustrated the use of SLiDe and its three main modules: Generator, Recogniser and Incremental Situator to develop multiple representations as a platform to learn from; construct observations from these representations; and learn the situatedness of shape semantics. From the five sets of observations constructed from five sets of multiple representations accumulating to 39 representations, the situatedness of recognised shape semantics are summarised in Figure 6.21 in the form of 8 situational categories shown in the graph in Figure 6.17. In Figure 6.21, there are three kinds of situational categories: constructed (C_{s5}, C_{s6}, C_{s7} and C_{s8}), reinforced ($*C_{s2}$, $*C_{s3}$ and $*C_{s4}$) and reconstructed ($C_{s1(a)}$). The extended control algorithm (read-evaluate-learn-trim) using the queue as discussed in Section 5.4.2.4 in the incremental clustering mechanism used in the Incremental Situator manipulates internally the decay of observations by deleting the ones with the lowest performance index off the queue. The shape semantics within each situational category are interlinked based on the situations within which they were recognised. Within each situational category, if a certain shape semantic were chosen to be the knowledge in focus, then the remaining shape semantics form the situatedness of that shape semantic and have the potential to guide its use based upon that within which it was learned previously. At the same time, within each situational category there is the possibility of the duality between knowledge in focus and other entities within the same situational category. The results within the group of learned situational categories is that for one knowledge in focus there might be a number of possible situations within which it could be recognised. Such possibilities allow different alternatives for the designers to choose from based on their interests. The chosen situation would be the one to guide the use of knowledge in focus while pursuing it further in designing.

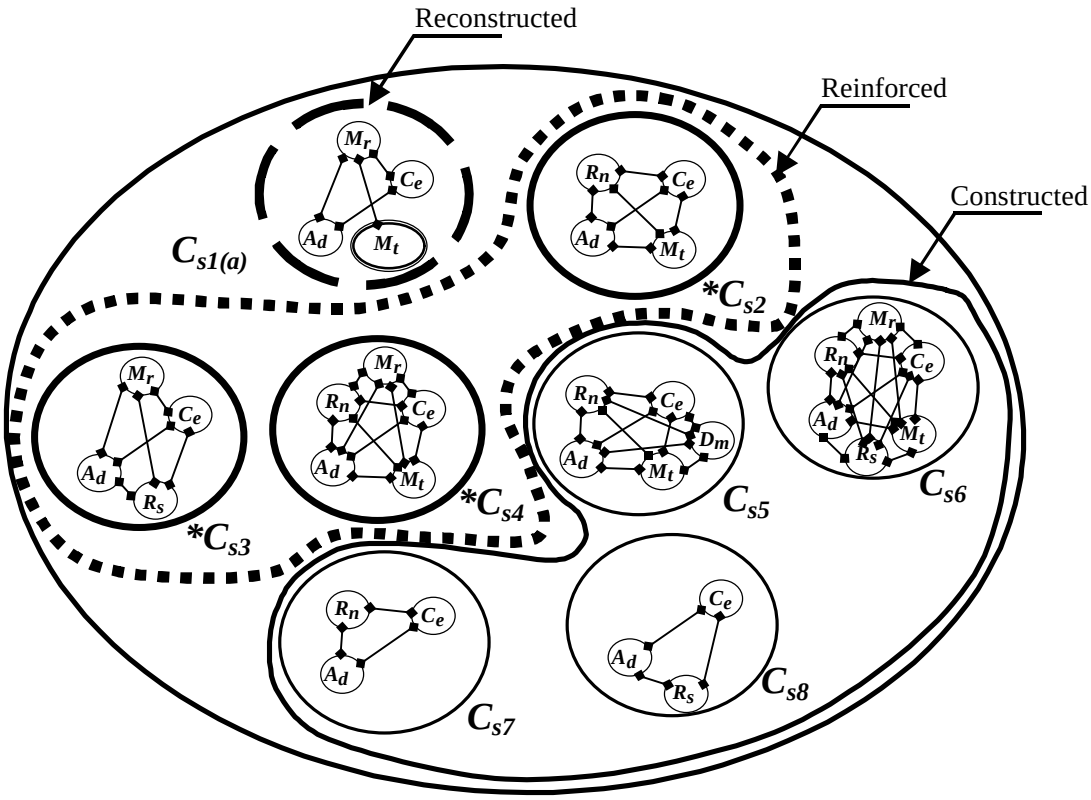


Figure 6.21 Three types of situational categories: constructed, reinforced and reconstructed, within the group of learned situational categories from the five sets of observations constructed from 39 representations of the design composition.

Chapter 7

SLiDe in Architectural Designing

This Chapter describes some ways in which SLiDe provides interactive support in designing compositions of architectural shapes. SLiDe-CAAD is proposed to help in enhancing the perceptual interaction with design elements in design composition; explore the design space for various alternatives and to help maintain the integrity of shape semantics of interest in the design composition. These could be achieved by the use of both necessary and sufficient conditions (predetermined), and the applicability conditions, situated and learned while designing. SLiDe-CAAD is introduced to offer a collaboration between the designer and the computer during the process of designing and producing a design artefact at the very early stages of designing.

7.1 Introduction

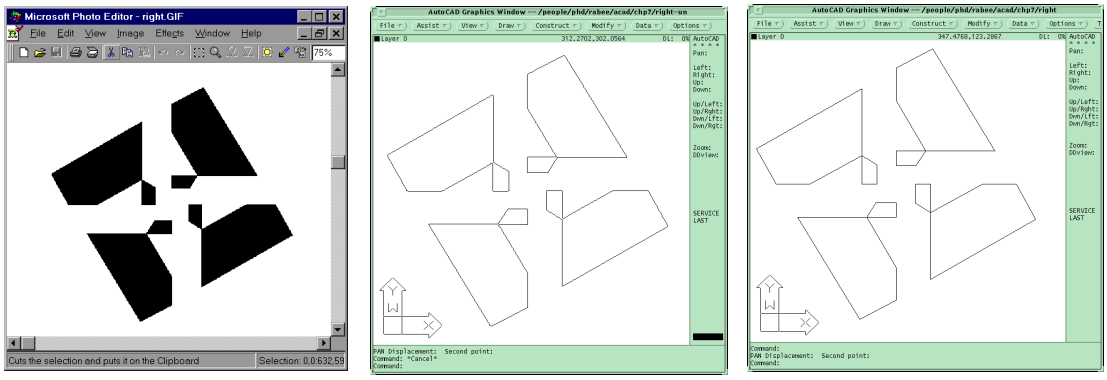
In architectural designing most of the current CAD systems can only be used at the late stages of the design process after most of the major design decisions have been made; and very few can be used during the conceptual stages of designing. During the process of designing, design solutions are fluid and emergent entities generated by dynamic and situated designing activities. Although designers have always managed with pencil, paper and imagination, computers could help them in designing (Stiny, 1990a). Designers work with descriptions involving drawings in many different ways. The ability to provide useful designing support at the conceptual stages of designing to accommodate the situated and fluid nature of early schematic designing and for design solutions to emerge is important. This Chapter introduces some ways in which SLiDe could be used to achieve this goal. Integrating SLiDe with current conventional CAD systems (SLiDe-CAAD), within the domain of designing architectural shapes could help with providing interactive computational support in designing at the early conceptual stages. The aims of SLiDe-CAAD are to provide a medium for the designers to explore the design space, to enhance the perceptual interaction with design elements and to maintain the integrity of developed design concepts based on designers' interest.

Within the proposed view of providing interactive support in designing, it is not meant to automate the design process but rather to help designers in designing. There are various scenarios for controlling such interaction. Chase (2000) introduced various possibilities of control strategies of the interaction between designer and computer that vary from full control to either the designer or the computer. Partial or major control could be assigned to one over the other. In this thesis, the role of the computer is to help

the designer in exploring different alternatives from which the designer may select a new move to develop further. At the same time, using the Recogniser module of SLiDe could provide CAD systems with the capability to recognise shape semantics in what have been developed and bring that to the attention of the designer. The designer may select one of the recognised shape semantics based on their interest. SLiDe-CAAD is proposed to explore the design space in the view of the desired shape semantic. This could help with enhancing the perceptual interaction with design elements in the design composition.

7.2 Enhancing the Perceptual Interaction with the Design Composition

During designing, designers usually develop their ideas in sketches. Design sketches are not something given to the designers at the beginning of the task, but something which designers dynamically produce from scratch during the design process (Suwa et al., 1998a). SLiDe's input can be either a design sketch (linear shapes drawn manually), or a drawing (linear shapes depicted in a conventional CAD system). Design sketches can be converted to a vectorised version of the one produced using conventional CAD systems. This process includes converting raster graphics to vector graphics. The initial data of this process is an image. A scanned image in a raster graphics format such as GIF or JPEG is transformed into the vector graphics format in DXF. In order to achieve this vectorisation process the KVEC software is used to convert GIF to DXF (http://ourworld.compuserve.com/homepages/kkuhl/). For instance, an image of a design sketch is shown in Figure 7.1(a), and the result of the vectorisation process is shown in Figure 7.1(b). The output image in vector graphics is processed to clear noise and to identify edge segments so that the expected picture of the image is achieved as shown in Figure 7.1(c).



(a) raster graphic scanned image (b) vectorised contour from the image (c) processed drawing cleared of noise

Figure 7.1 Conversion of a design sketch to a vectorised version that can be handled by CAD systems.

Visual perception requires the capacity to extract shape properties and relationships among shape parts. SLiDe-CAAD could help with enhancing the perceptual interaction between the designer and the design composition. SLiDe-CAAD can help in exploring the design space and bringing designer's attention to a set of congruent shapes derivable

from their current design composition, by highlighting them and reflecting certain shape semantics of interest to the designers. The proposed processes to achieve this goal are indicated in Figure 7.2. The recognised shapes might attract a designer's attention to pursue them further in designing. This implies leading designers to unintended moves.

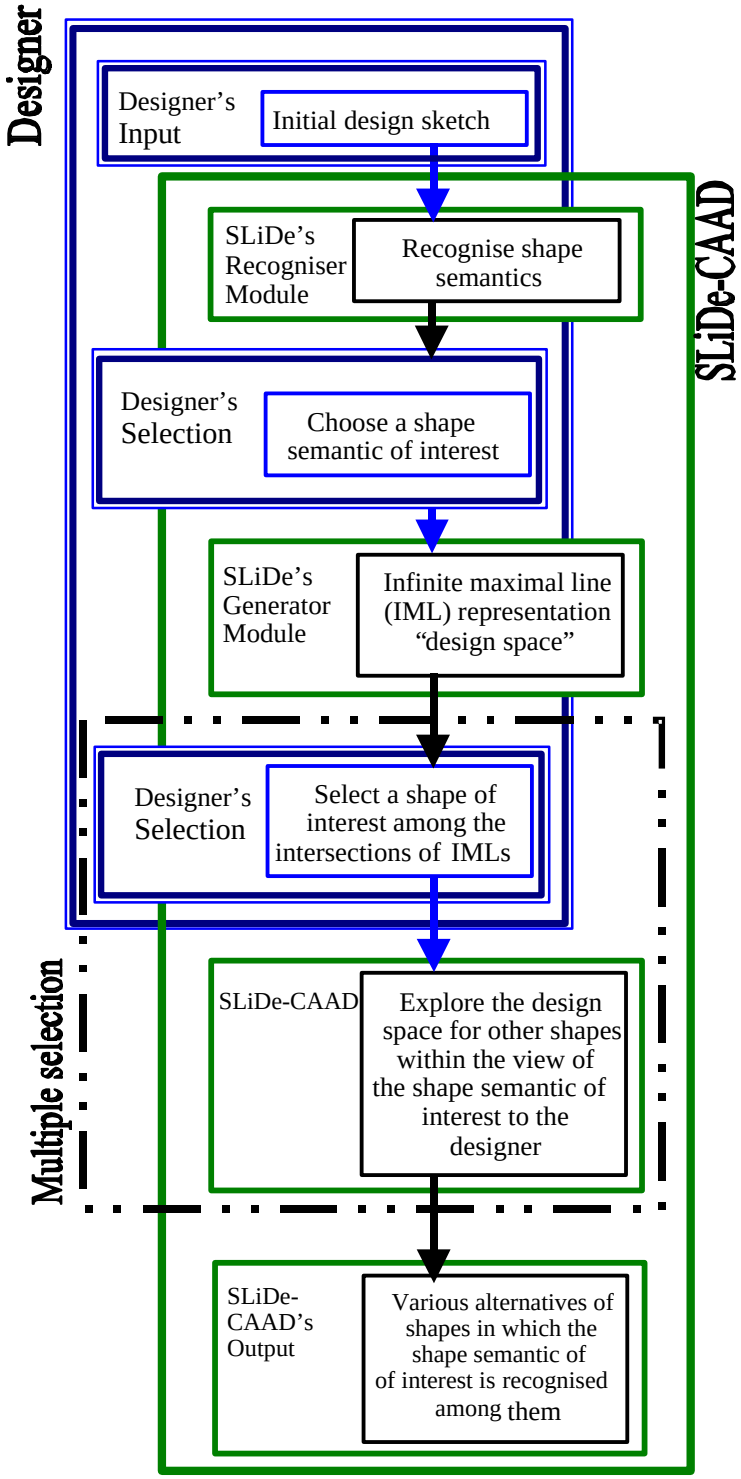
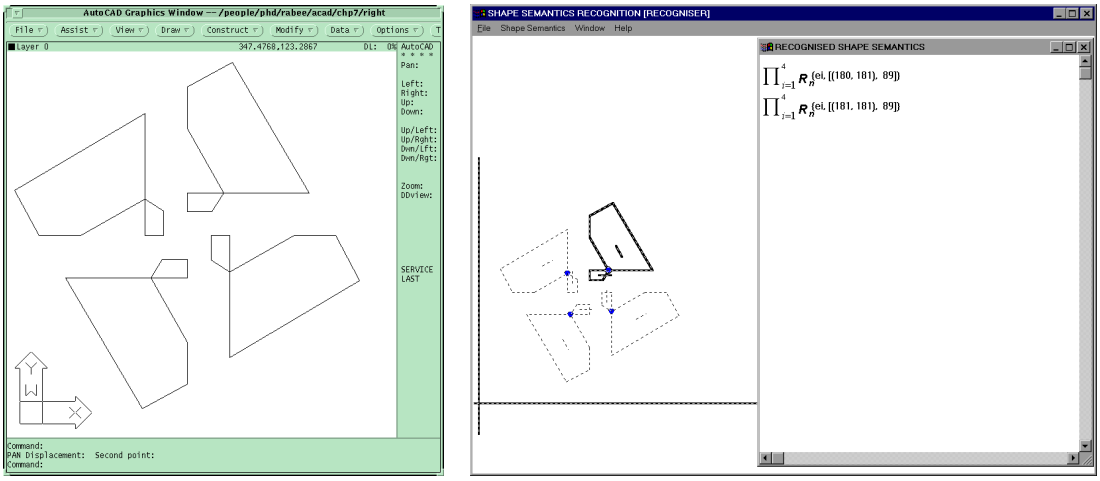


Figure 7.2 A framework for enhancing the perceptual interaction with the design composition.

The initial design sketch (design composition), is converted into a drawing that can be dealt with in conventional CAD systems as shown in Figure 7.3(a). The Recogniser module of SLiDe can be used to detect shape semantics in the design composition. The result of using the Recogniser module is shown in Figure 7.3(b) whereby a cyclic rotation ($\boldsymbol{R}_n$) is found. Designers may indicate their interest in one of the recognised shape semantics found by the Recogniser module in their initial design composition. SLiDe-CAAD could help in exploring the design space to find other shapes within which the desired shape semantic is recognised. The design space of the initial design composition is constructed using the Generator module to develop infinite maximal lines of the initial design composition as shown in Figure 7.4(a). The design space can be searched for other congruent shapes that satisfy the desired shape semantic. This search is guided by both the shape semantic of interest to the designer and the selected shape from amongst the intersections of infinite maximal lines as shown in Figure 7.4(b). The recognised congruent shapes in the view of cyclic rotation are highlighted in shaded areas to be easily seen by the designer as shown in Figure 7.4(c).



(a) (b)

Figure 7.3 (a) initial design composition; (b) the result of using the Recogniser module to detect shape semantics in the initial design composition

Designers may continue using SLiDe-CAAD to explore the design space for other shapes that satisfy shape semantics of interest to them, eg. cyclic rotation in relation to other shapes. Figures 4.7(d), (e) and (f) show some examples of new congruent shapes wherein cyclic rotation is recognised. This provides designers with rich alternatives from which to choose in further pursuing in designing. As it can be seen from these alternatives, the use of guided search in SLiDe-CAAD could help in making implicit shapes in the initial design composition explicit and perceivable by the designers based on their interest in relation to a certain shape semantic.

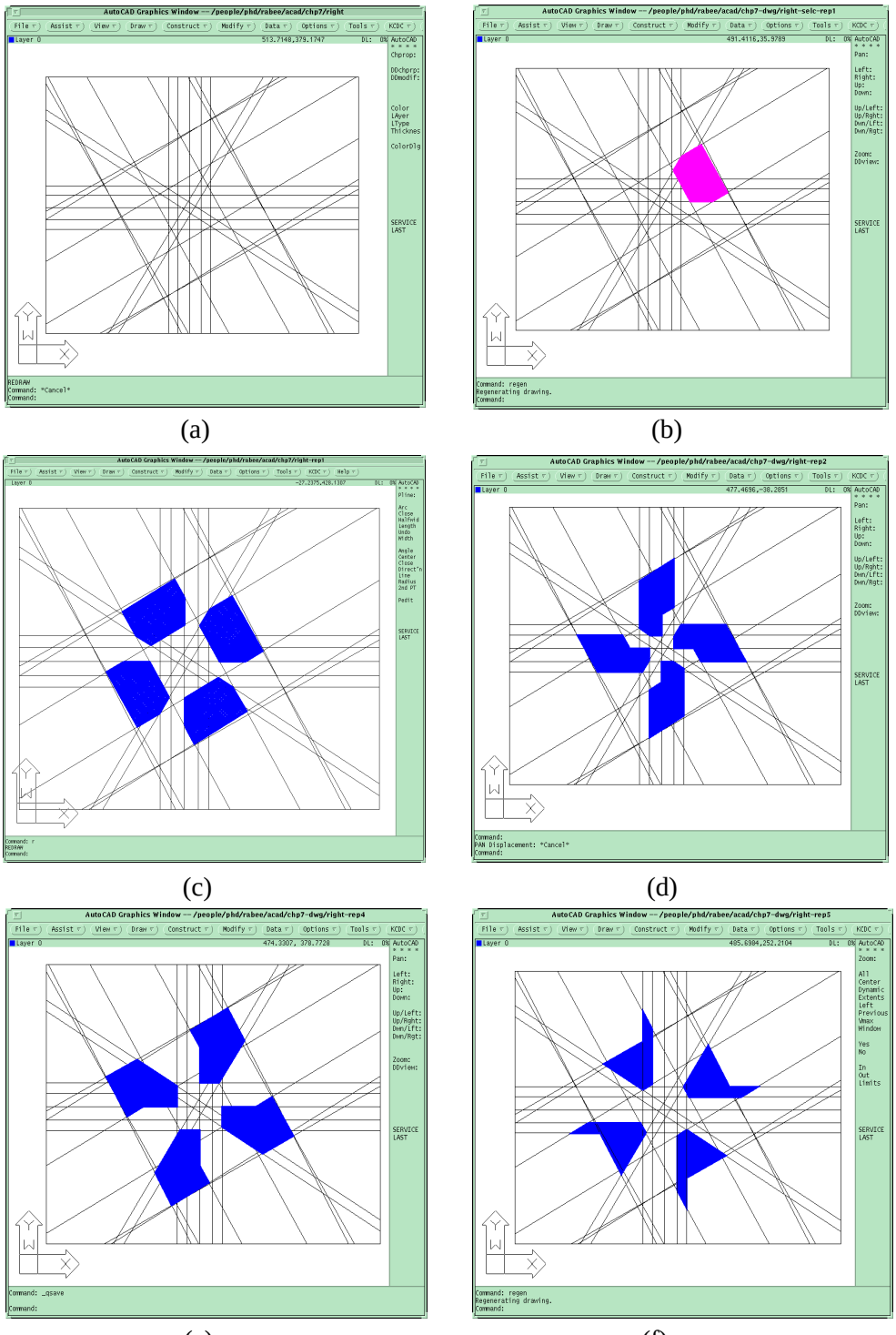


Figure 7.4 (a) the design space in the form of the intersections of infinite maximal lines of the initial design composition within the frame drawn by the designer; (b) a selected shape by the designer to be used in searching the design space for other congruent shapes that satisfy cyclic rotation; (c) a group of congruent shapes among which cyclic rotation is recognised; (d), (e) and (f) other examples of congruent shapes that satisfy the designer's interest in cyclic rotation.

7.3 Exploring various alternatives in the design space

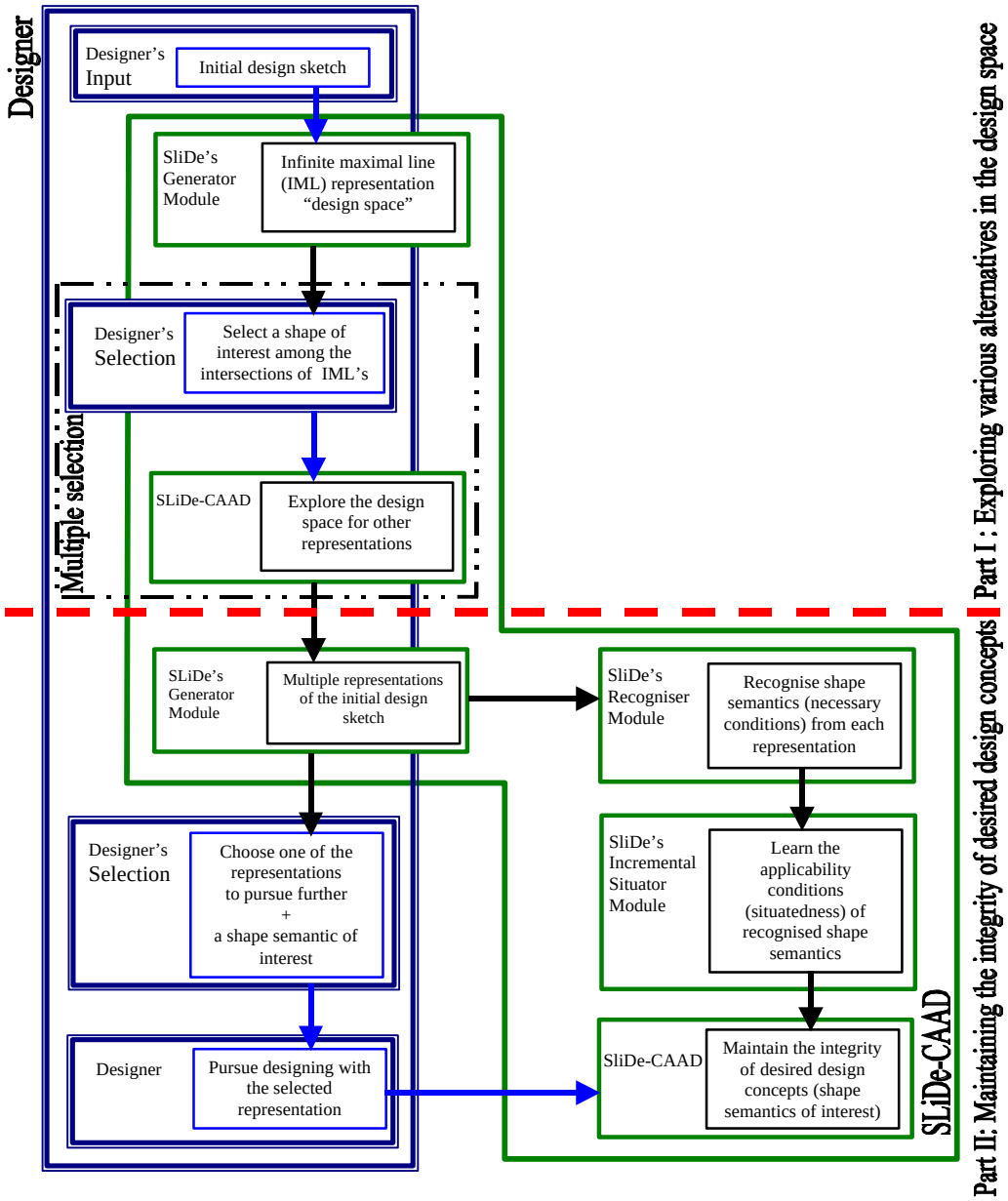
The use of multiple representations, provided by the Generator module in SLiDe, could be useful for designers in conceptualising, exploring and perceiving their designs differently. This helps in exploring the shapes in a design composition and enabling designers to have a variety of representations of what has been designed which may lead them to different discoveries from those they may otherwise have pursued. These different representations of the same design composition help to focus the designer's attention on hidden features of their design elements. This is achieved through the use of the Generator module in SLiDe as shown in Part I in Figure 7.5. The Generator constructs the design space by developing infinite maximal lines of the initial design composition. The designer selects a shape of interest from among the intersections of the infinite maximal lines. The Generator searches the design space for congruent shapes of the selected shape. The congruent shapes are highlighted by shading the shapes and the Generator module develops a representation from the recognised congruent shapes and the leftover of the initial design composition. Designers may continue exploring the design space for other alternatives by selecting other shapes of interest from among the intersections of infinite maximal lines and consequently the Generator module develops a corresponding representation. Examples of some of the alternate representations developed using the Generator module are shown in Figures 7.6 (a) to (f) showing the representations N_1 to N_6.

7.4 Maintaining the Integrity of Desired Design Concepts while Designing

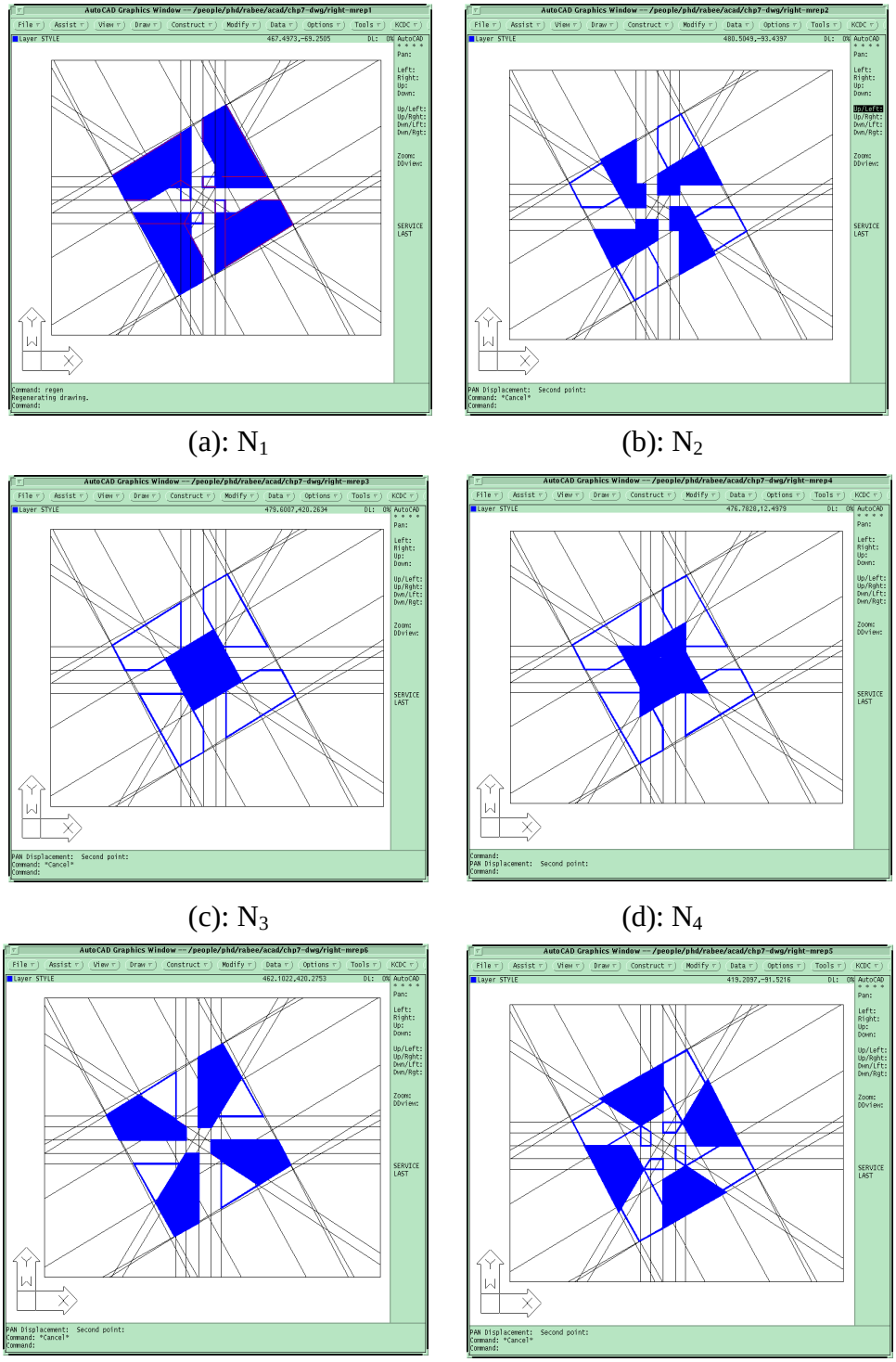
After a number of design alternatives have been generated in the exploration phase, the designer may select one of the alternatives to pursue further. A designer might be attracted to a specific design alternative for different reasons. One of these could be a certain shape semantic. During the conceptual designing process, designers usually revise the selected alternative by making changes while designing. There are two classes of possible changes: addition and substitution (Gero, 1992a). In the context of shape composition, the concept of addition is that shape parts are added to the existing stock of shapes which are used to describe the design composition. On the other hand, the concept of substitution is that some existing shapes in the design composition are deleted and others are amended to produce different sets of possible design compositions. It might be useful to provide designers with a tool that has the capability to maintain consistency in shape semantics when changes are made to the concepts that have attracted them in their designs at earlier stages. Maintaining the integrity of desired and developed concepts while designing is another way in which SLiDe-CAAD could provide support in architectural designing. A simplified approach to maintain the shape semantic of interest in the design composition has been proposed by Gero and Jun (1995b). The shape semantic that has been selected to be kept is constrained to exist independently of other operations based on the sufficient and necessary conditions of that shape semantic, ie situation independent. It is proposed here to maintain the desired shape semantic by preserving both its necessary conditions (predetermined) and situatedness (constructed). This would help not only to maintain the shape semantic of

interest but also its situation. Maintaining the situation within which the shape semantic of interest was recognised helps with maintaining the integrity among shapes in the design composition as a whole. This means that the desired shape semantic is constrained by both necessary and applicability conditions. This is to be achieved through maintaining the other shape semantics within which the shape semantic of interest is situated as outlined in Part II, Figure 7.5.

Figure 7.5 Part I: Framework of exploring various alternatives in the design space and Part II:



Framework of maintaining the integrity of desired design concept, shape semantic of interest, by preserving both of its necessary and applicability conditions.



(a): N_1 (b): N_2

(c): N_3 (d): N_4

(e): N_5 (f): N_6

Figure 7.6 Various alternatives from exploring the design space of the initial design composition using the Generator module, from (a) to (f) show the representations N_1 to N_6.

In Figure 7.4, the Recogniser module of SLiDe can be used to detect shape semantics at each developed representation generated while exploring the design space as shown in

Figure 7.6. The result of using the Recogniser module is a set of observations constructed as shown in Table 7.1 from the developed representations. The Incremental Situator module of SLiDe can be used to learn the situatedness of recognised shape semantics across the observations constructed using the Recogniser module. The results of using the Incremental Situator module are situational categories as shown in Figure 7.7. These situational categories are used to define the applicability conditions of recognised shape semantics.

Table 7.1 A set of observations produced using the Recogniser module to detect shape semantics in the developed representations shown in Figures 7.6.

Representation No.	Corresponding Observation (O_n)	
N_1	O_1	$\boldsymbol{R_n, A_d}$
N_2	O_2	$\boldsymbol{R_n, A_d}$
N_3	O_3	$\boldsymbol{R_n, C_e, A_d}$
N_4	O_4	$\boldsymbol{R_n, C_e, A_d}$
N_5	O_5	$\boldsymbol{R_n, A_d}$
N_6	O_6	$\boldsymbol{R_n, A_d}$

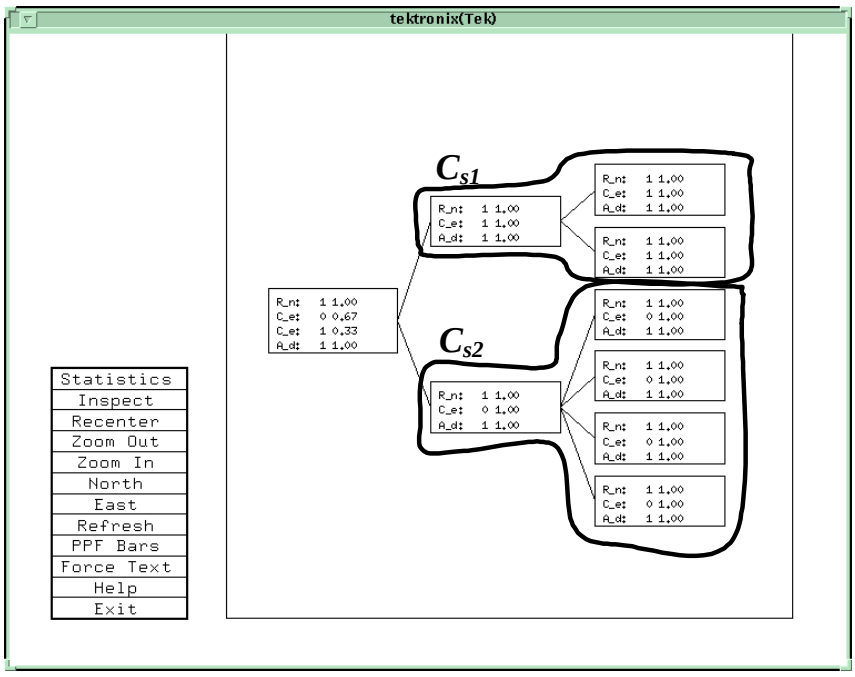


Figure 7.7 The result of using the Incremental Situator module in SLiDe to learn the applicability conditions of recognised shape semantics across the observations wherein two situational categories $\boldsymbol{C_{s1}}$ and $\boldsymbol{C_{s2}}$ are learned.

Within each situational category, if a certain shape semantic were selected to be the knowledge in focus the remaining shape semantics within this category form the situatedness of that shape semantic. In Figure 7.7, there are two learned situational categories: $\boldsymbol{C_{s1}}$ and $\boldsymbol{C_{s2}}$. In $\boldsymbol{C_{s1}}$, there is regularity among the shape semantics $\boldsymbol{R_n}$, $\boldsymbol{C_e}$ and

A_d that refer to cyclic rotation, centrality and adjacency respectively. Within C_{s1}, if cyclic rotation (R_n) is selected to be the knowledge in focus, then both centrality (C_e) and adjacency (A_d) are the applicability conditions of cyclic rotation. In other words, cyclic rotation is situated within centrality and adjacency. In C_{s2}, there is another situation wherein cyclic rotation is recognised in conjunction with adjacency.

The situatedness of shape semantics could be used to maintain shape semantics and the integrity of the design composition while revising the design. SLiDe-CAAD could help to dynamically change the association between the parts in a design composition based upon the designers' selections of a shape semantic of interest by maintaining its applicability conditions. The selected shape semantic of interest to the designer is considered by SLiDe-CAAD as the knowledge in focus or a desired design concept. So, whenever designers modify their designs, SLiDe-CAAD would automatically maintain the integrity of their desired design concepts through maintaining the related shape semantics that define the situation of the desired design concept.

7.4.1 Maintaining the integrity of design concepts in response to addition of shapes

For instance, let us assume that the designer, after exploring the design space, using the Generator module of SLiDe to develop some representations, selected one of the new developed representations, eg. representation N_3 as shown in Figure 7.6(c), as a new move to further pursue in designing. This selected representation is now the current design composition that the designer acts on as shown in Figure 7.8(a) wherein the designer is interested in cyclic rotation (R_n) among the group of four shapes S_1 in the design composition. Some time later, the designer decided to add or insert a space, in the form of a new shape S_3, between the two shapes S_1 and S_2 in the design composition as shown in Figure 7.8(b). Such addition required moving the shape S_1 from its previous location. Hence, the cyclic rotation (R_n) among the group of shapes S_1 is disturbed by moving one of the shapes S_1 and changes its distance from the rotation centre of its group. Yet, there is a possibility to maintain the cyclic rotation by moving the other shapes S_1 with the new distance from the rotation centre as shown in Figure 7.8(c). Maintaining the distance between each of the congruent shapes S_1 and the rotation centre is one of the necessary and sufficient conditions of cyclic rotation. In spite of maintaining the cyclic rotation in the design composition, the adjacency (A_d) between each of the shapes S_1 and S_2 is disturbed. From the learned situational category (C_{s1}) in SLiDe, adjacency (A_d) is one of the applicability conditions of cyclic rotation. Since, cyclic rotation is the knowledge in focus, SLiDe-CAAD could help in maintaining the situatedness of cyclic rotation via preserving all of its applicability conditions. As a result, the shape S_3 is inserted between each of the shapes S_1 and S_2 to maintain the adjacency among them as shown in Figure 7.8(d) whereby adjacency is maintained in one of its alternate forms from direct contact to a link between shapes. As can be seen from this example, maintaining both the necessary (preconditions) and applicability conditions (situatedness) provide a rich support to maintain the integrity in a design composition as a whole. The necessary and sufficient conditions could be used to maintain the shape semantics and the applicability conditions to maintain the situatedness of shape semantics.

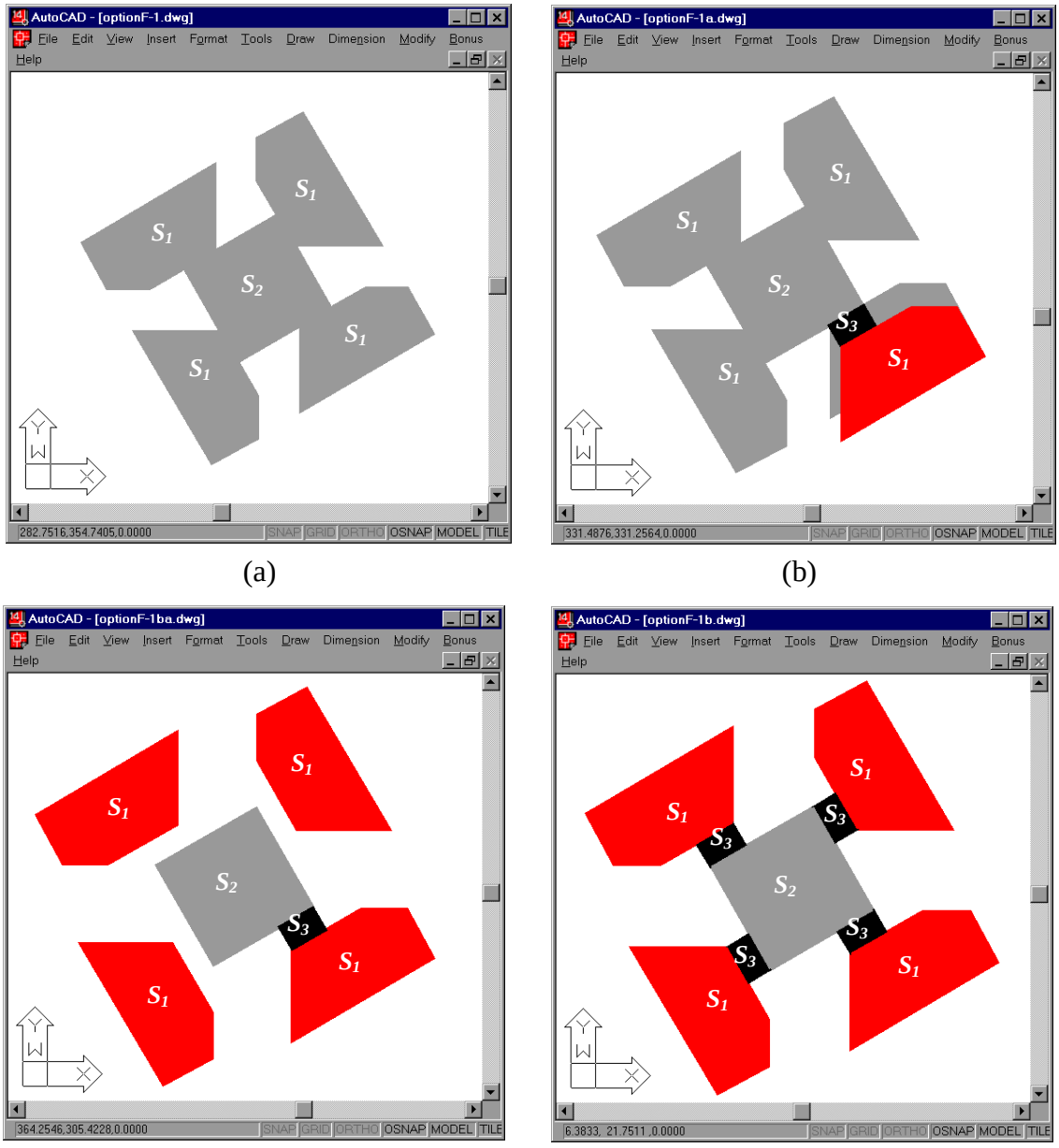
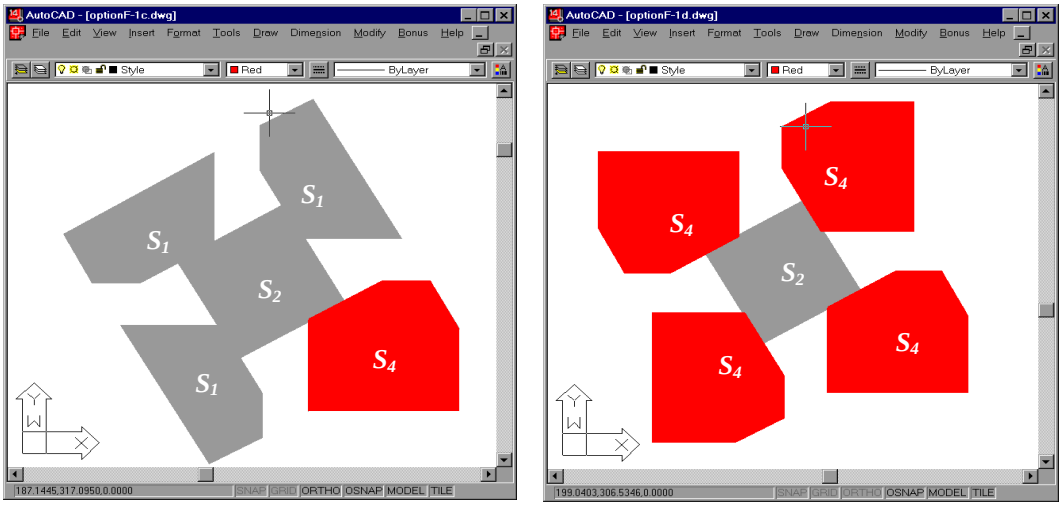


Figure 7.8 (a) A new move that a designer selected from the developed representations to further pursue in designing, (b) a new space added by the designer at a later stage, (c) SLiDe-CAAD could help in maintaining the integrity of cyclic rotation via preserving its necessary conditions, and (d) SLiDe-CAAD could help in maintaining the situatedness of cyclic rotation via preserving its applicability conditions, eg. adjacency and centrality.

7.4.2 Maintaining the integrity of design concepts in response to substitution of shapes

The designer may amend any of the shapes in the design composition substituting an existing shape as shown in Figure 7.9(a) whereby the new shape S_4 substitutes the existing shape S_1. The substitution of the existing shape disturbs the cyclic rotation since the shape S_4 is not congruent to the shape S_1 where congruency among shapes is one of the necessary and sufficient conditions of cyclic rotation. SLiDe-CAAD could help in maintaining the cyclic rotation by substituting each of the S_1 shapes with S_4 as shown in Figure 7.9(b). In response to this change SLiDe-CADD will check out all the learned applicability conditions of cyclic rotation: adjacency and centrality to maintain its situatedness. In this example, neither adjacency nor centrality was disturbed.

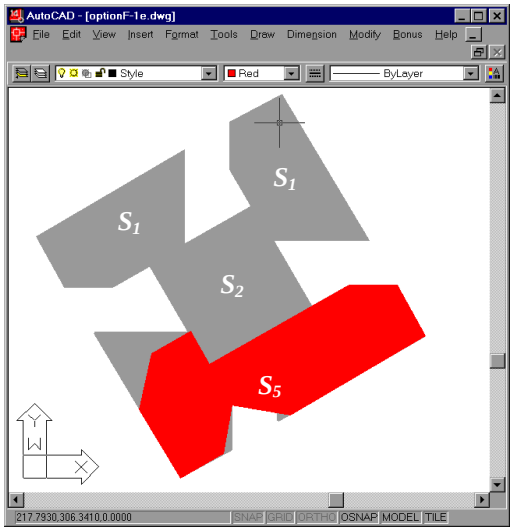


(a) (b)

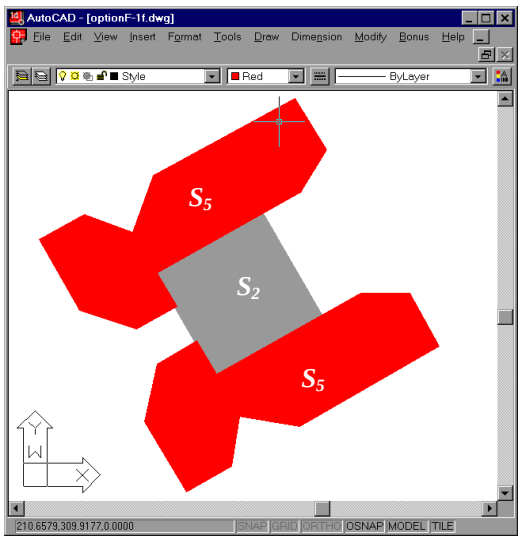
Figure 7.9 (a) Substituting one of the existing shapes (S_1) with a new shape (S_4), (b) maintaining the cyclic rotation via preserving the congruency among shapes and substituting each of (S_1) with (S_4).

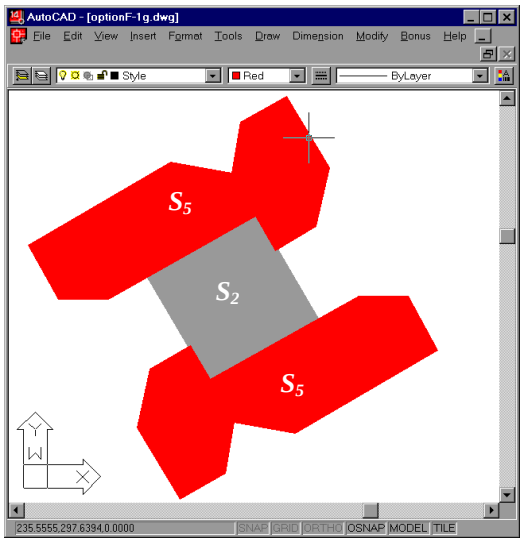
On the other hand, there are various kinds of change that the designer may endeavour to achieve during designing. An example of such endeavour might be similar to the one shown in Figure 7.10(a) whereby the designer decided to join two of the (S_1) shapes together with some modifications forming a new shape (S_5) that substitutes them. With such change, the cyclic rotation is not only disturbed but also violates the constraints within which there is no possibility to maintain the necessary and sufficient conditions of cyclic rotation within that design composition since the overlap among shapes is not permitted. Nevertheless, there are some other possibilities in which SLiDe-CAAD could help designers in such a case. One of these is giving the designer the possibility of changing the shape semantic of interest. SLiDe-CAAD then examines the possibility of accommodating the new shape semantic as the knowledge in focus within the design composition by trying to satisfy both of its necessary and applicability conditions and if it is successful highlights the results visually to the designer. For instance, Figure 7.10(b) shows the expected results of changing the knowledge in focus from cyclic rotation to reflective symmetry and in Figure 7.10(c) to simple rotation. On the other

hand, changing the shape semantic of interest could be fully automated in SLiDe-CAAD to provide designers with the successful alternatives of changing the focus. The automated selection process could be triggered by the violation of the constraints of the existing shape semantic of interest. The evaluation of successful alternatives of changing the focus is measured by the ability to satisfy both the necessary and applicability conditions of the new shape semantic within that design composition.



(a)





(b) (c)

Figure 7.10 (a) Joining and modifying two of the existing shapes (S_1) to form a new shape (S_5), (b) and (c) the expected results of changing the shape semantic of interest from cyclic rotation to reflective symmetry and simple rotations respectively.

7.5 Discussion

This Chapter outlined some features that SLiDe-CAAD could offer to conventional CAD systems in enriching both interactivity and designing support in the preliminary stages of designing. These features might help with enhancing the perceptual interaction with design elements in the design composition; explore the design space for various alternatives and maintain the integrity of shape semantics of interest in the design composition and its situatedness. The purpose of SLiDe-CAAD is not to replace the designer and automatically produce solutions to a shape problem that the designer has formulated, but to collaborate with the designer in designing and producing a solution. Considering the use of both necessary and applicability conditions to maintain the design integrity adds a further dimension to parametric design; that is, the situatedness of design knowledge. The main concern in parametric design is the imposition of constrained relationships (semantics), on the shape of objects, which enables shape manipulation by adjusting several geometrical attributes in some fixed relationship to each other or in a relationship to explicit changes applied to other shapes, or to the location of other objects (Kalay, 1989; Rossignac et al., 1989). Such constraints are predetermined and the main concern is to maintain a certain semantic in the shape regardless of its interdependency with other semantics. Such interdependency cannot be predefined but rather learned based on the situation within which that shape semantic is recognised. Furthermore, these interdependencies and relationships among shape semantics are changing based on the observations constructed from the various representations developed while designing. The Incremental Situator module in SLiDe could be used to capture the effect of such a change in the situatedness of recognised shape semantics. These features provide the potential to change the nature of passive conventional CAD systems to be active and responsive CAAD support systems at the very early stages of designing.

Chapter 8

Conclusion

This chapter summarises the research results of this thesis. The contributions of this research and possible directions for future research are outlined.

This thesis has presented an approach to *situated learning in designing* based on the notion that both learning and designing are *situated* and learning design knowledge in relation to its *situation* is more useful than learning it independently of its locus and application. The concept of *situatedness* acknowledges that where what is done and when, matters. The approach of *situated learning in designing* is to focus on constructing a conjunction between design knowledge and the situation within which it was recognised. The situatedness of design knowledge is perceived as the applicability conditions that have the potential to guide the use of design knowledge. In this thesis, a computational system for *situated learning in designing* (SLiDe) was conceived, developed and implemented within the domain of *architectural shape semantics*. SLiDe is an acronym coined in this thesis. The *situations* within which *shape semantics* were recognised from a set of observations in a design composition were organised into the form of a hierarchical tree structure. These observations were constructed from a set of multiple representations that were generated from a single design composition. The multiple representations allowed for various kinds of shape semantics and relationships among those shape semantics to be recognised. The situatedness of shape semantics was constructed from the regularities of the relationships among shape semantics across the observations within which those shape semantics were recognised. The constructed situations and their hierarchical structure tree change in response to new observations. SLiDe has the capability to learn and to refine the situatedness of shape semantics incrementally. In this Conclusion contributions of this research are presented and possible directions for future related research are suggested.

8.1 Objectives and Results

The aim of this thesis was to develop an approach to situated learning in designing in which computers would have the capability to learn design knowledge including concepts of the situatedness of design knowledge. Architectural shape semantics were proposed to be the domain of applying situated learning in designing. Four main objectives were proposed in order to achieve this aim: development of multiple representations; learning of shape semantics in relation to their situations; development

of a computational system of situated learning in designing (SLiDe); and exploration of the application of SLiDe in architectural shape designing. Research results are summarised in view of these objectives as follows.

(i) The objective of developing multiple representations from a single design composition (floor plan), to serve as a platform for a situated learning system in designing was achieved by developing an infinite maximal lines representation of the initial shape of the design composition and developing the Generator module that has the capability to be used with a designer's interaction for generating various representations from a single design composition.

(ii) The objective of learning shape semantics in relation to the situations within which they were recognised was achieved by developing the Recogniser module that is able to detect shape semantics from each developed representation and construct a corresponding observation from each representation. The Situator module learns shape semantics in relation to their situations through learning the regularities of relationships among shape semantics across the observations within which they were recognised.

(iii) The objective of developing a computational system for situated learning in designing was achieved by developing SLiDe that has the capability to: develop multiple representations; recognise shape semantics from each developed representation; construct sets of observations from the developed representations; learn the situatedness of recognised shape semantics; and incrementally update what has been learned in response to new observations using the Reconstructing Situator in the Incremental Situator module.

(iv) The objective of exploring ways of using SLiDe in architectural shape designing was achieved by demonstrating some of SLiDe's capabilities integrated with CAD systems within the architectural domain (SLiDe-CAAD), to enhance the perceptual interaction with design elements to bring designers' attention to hidden visual features in their designs; explore the design space for various alternatives; and maintain the integrity of shape semantics of interest and their situatedness.

8.2 Contributions

The contributions of this thesis are:

- ***Multiple representations of a design composition*** - A design composition (shape of a floor plan), was interpreted in various ways in which different shape semantics and relationships among them could be recognised from the representations. The use of multiple representations in designing allowed for the transformations of a design between designing states. The use of infinite maximal lines provided a rich repertoire within which multiple representations were developed. The concept of using multiple representations to encounter various

states of situatedness was implemented. The development of multiple representations achieved using the Generator module in SLiDe served as a foundation upon which many kinds of inference were developed. At each representation there were opportunities to recognise various shape semantics and reinforce or decay their situations. A learning system may identify the regularities of relationships among recognised shape semantics to learn within which situations those shape semantics were recognised.

- ***Shape semantics recognition*** - The recognition of shape semantics is a kind of appreciation of semantics in design compositions. Shape semantics are recognised in terms of similarity of spatial relationships and physical properties. The identification of shape congruency is an important constituent for recognising some of the shape semantics. Building on the work of SPARS (Cha and Gero, 1998), the Recogniser module developed in SLiDe helps with endowing the computer with the ability to appreciate the shape semantics within each developed representation through recognising various sets of shape semantics. Three sets of shape semantics that are among the most prominent semantics in architectural shape composition can be recognised using the Recogniser module: expression; symmetry; and modality. Expression includes dominance and adjacency; symmetry includes reflective symmetry around one axis or around more than one axis, simple or cyclic rotation, translational repetition and scaling; and modality includes centrality, radiality and linearity. The Recogniser module has the capability to detect each representation against these sets of shape semantics.

- ***Learning the situatedness of recognised shape semantics*** – The acquisition of design knowledge is a capability constructed in action within the situation. This leads to the notion of situatedness to provide the potential for guiding the use of knowledge. The concept of situatedness is borrowed from situated cognition that asserts “where you are when you do what you do matters”. Since it is not possible in designing to know beforehand what knowledge to use in relation to any situation, then there is a need to learn knowledge in relation to its situation within which it was recognised. The Situator module developed in SLiDe has the capability to learn the regularities of relationships among shape semantics across the observations produced using the Recogniser module. The Situator module clusters these regularities in the form of situational categories in a hierarchical structure tree. The learned situational categories carry with them the applicability conditions of recognised shape semantics. Within each learned category, if a certain shape semantic were chosen to be the design knowledge in focus, the remaining shape semantics within this category form the applicability conditions of this shape semantic within which it was recognised across the observations. The duality between knowledge in focus and its situation has been exploited. This provided the opportunity to learn the mapping from one to many in which there might exist a number of situations within which the design knowledge in focus could be recognised. This enriches the relationships between design knowledge and its states of situatedness.

- ***Incremental learning of the situatedness of recognised shape semantics*** – During the process of designing the relationships between knowledge and situation are not treated as static, but are invariably subject to change. This is due to the processes of restructuring and reinterpretation. Once the situations have been constructed using the Situator module, they can be updated. The incremental process of learning the situatedness of recognised shape semantics involves: constructing new regularities of relationships among shape semantics; reconstructing learned regularities to include new shape semantics that are relevant to a learned situation; reinforcing learned regularities to empower what has been learned; and decaying learned regularities to decline some of what has been learned. The observations that SLiDe learns from might be constructed at different points of time and usually not in order of their importance. Incremental learning of the situatedness of shape semantics is achieved by developing the Incremental Situator module in SLiDe in which the concept drift is considered to almost eliminate the effect of the order of observations in the learned situational categories. Incrementally updating the situatedness of what has been recognised makes SLiDe sensitive to the new observations whenever they are constructed.

- ***The utility of SLiDe in architectural designing*** – Conventional CAD systems can only be used at the late stages of designing when most of the conceptual designing issues have been concluded. Integrating SLiDe with conventional CAD systems in the domain of architectural shape composition (SLiDe-CAAD), could help with providing interactive support in designing at the early stages. SLiDe-CAAD could enhance the perceptual interaction with design composition in which it can help in exploring the design space and bringing designers' attention to a set of congruent shapes derivable from their current design composition, by highlighting them and reflecting certain shape semantics of interest to designers. The use of multiple representations, provided by the Generator module in SLiDe, could be useful for designers to conceptualise, explore and perceive their designs differently. This helps to explore the shapes in a design composition and allows designers to have a variety of representations of what has been designed in which it may lead them to different discoveries from those they may otherwise have pursued. The applicability conditions for each recognised shape semantic, learned using the Incremental Situator module in SLiDe, could be facilitated to maintain its situatedness in the design composition as well as the integrity of the design composition as a whole while refining the design. SLiDe-CAAD could help to dynamically change the association between the parts in a design composition based upon the designers' selections of a shape semantic of interest by maintaining its applicability conditions. The necessary and sufficient conditions predefined in the Recogniser module could be used to maintain the integrity of the shape semantic of interest in particular while refining the broader design composition.

- ***Definition of future related research*** – The concepts developed and the results achieved in this thesis led to the definition of future related research as outlined in section 8.3.

8.3 Future Work

The computational system of situated learning in designing developed in this thesis is an initial step to capture the situatedness of design knowledge while designing. The concept of situatedness is not limited to what has been explored in this thesis. It is far richer in its potential in which further concepts could be included within the situatedness of design knowledge. More experience and additional research are required to realise the ultimate goal of this research. This research can be extended in the following directions.

8.3.1 Situated learning within the context of use while designing

The framework of situated learning in designing developed in this thesis can be extended to construct the situatedness of design knowledge within the context of use while designing. During the process of designing, a design solution is a fluid emergent entity generated by dynamic activities in which design elements change over time according to changes in the immediate context. The design solution is created by the designer's situated activities. So the situatedness of design knowledge could be constructed in a computer-based interaction from a continuous stream of designer's actions. The benefit of learning the situatedness within the context of use over learning the situatedness of design knowledge within which it was recognised (as it has been developed in SLiDe) is that the situatedness of design knowledge could be learned as it is encountered in action while designing. In designing, various transformations take place during the development process of reaching a design solution. Hence, the context changes when design transformations are applied to the initial design composition. So, alternatively to develop a set of multiple representations of a design composition from its infinite maximal lines representation to serve as a platform of SLiDe is to capture each set of transformations in action while designing and learning or updating the situatedness of design knowledge in response to them. Most importantly, is that a mapping could be drawn between designers' actions that led to certain transformations and their consequences on the situatedness of design knowledge. The change in SLiDe's framework would be as shown in Figure 8.1. The sequence of procedures for changing Slide's framework is as follows.

(i) Use the Recogniser module to detect shape semantics from the initial design composition and construct an observation.

(ii) Use the Situator module to learn the situatedness of recognised shape semantics based on their relationships in the first observation. The Situator module initialises its hierarchy to a single node basing the values of this category's attributes on the first observation

(iii) Observe designer's actions and capture both designer's actions and the current design composition whenever transformations take place as a new representation and use the Recogniser module to construct a corresponding observation.

(iv) Use the Restructuring Situator module upon encountering a new observation to update the initial hierarchy structure to accommodate the new observation through either incorporating it in the previous category or creating a new category.

(v) Learn the relationships between designer's actions and their effect on the situatedness of recognised shape semantics.

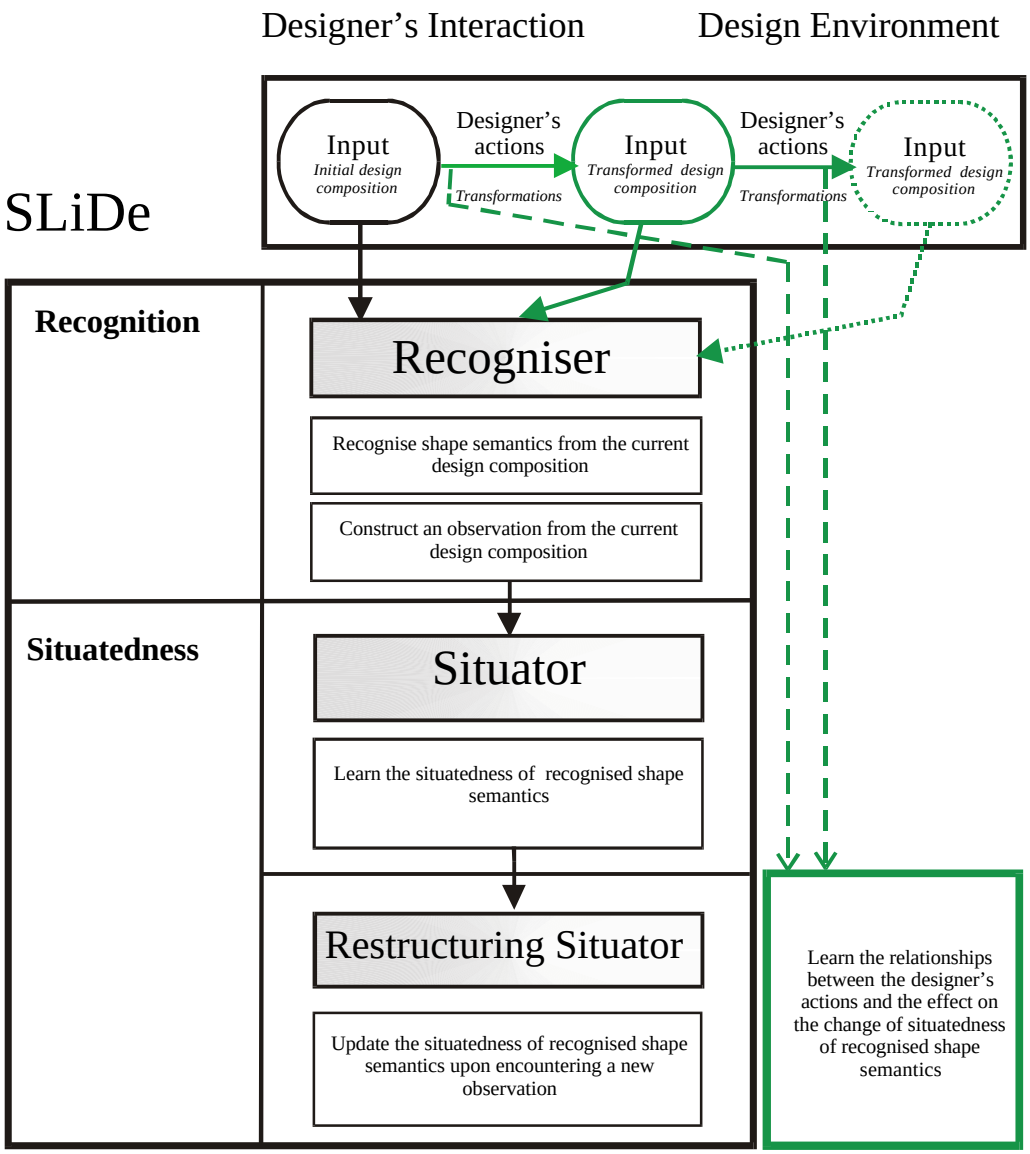


Figure 8.1 An initial framework of SLiDe's possible further development to learn within the context of use while designing.

8.3.2 Considering the role of functional knowledge while learning the situatedness

Designing includes transforming stated human needs into a design composition whose functions are to satisfy the required needs (Rosenman and Gero, 1998). Essentially a structural description of a design composition tells what it is, while a functional description tells what it intended to accomplish. That is, functional descriptions specify actions with effects that interest us (Mitchell, 1990). Another possible direction that is rather intriguing is to consider the relationships of function, structure and behaviour while constructing the situatedness of design knowledge. That is, learning the

situatedness of shape semantics recognised in an architectural shape composition is not only according to what the shapes in a design composition look like, but also according to their suitability. In a given situation, the effects of various actions performed by the designer could be sorted to "useful" or "irrelevant" based on the effects of these actions in satisfying the predefined functional requirements. The actions with useful effects could be regarded as goal oriented and therefore be classified based on that. Such classification is relative to a state of knowledge and a point of view. A claim that an action is useful invites a question about the viewpoint of considering emergent functions. Since in designing, neither the state space nor the goal state is fixed, then a reasonable way to consider them is to evaluate the usefulness of actions based on both explicit and emergent functions to consider possible uses that were not intended. Recent cognitive studies of designers' activities (Suwa et al., 1999) showed that designers were able to find important aspects of a given problem and thereby invent design issues or requirements during the process. They were able to invent design issues or requirements through interaction with sketches, sometimes by using explicitly articulatable knowledge, and sometimes by constructing justifiable reasons on the fly, while designing.

8.3.3 Using SLiDe within an autonomous situated agent-based designing system

An agent is a finite system with beliefs, goals and actions (Russell and Norvig, 1995). An autonomous agent is an agent capable of reacting to and reasoning about events that occur in its environment, executing actions in order to achieve goals in its environment and communicating with other agents (Das et al., 1997). For an agent to do so, it needs to construct its beliefs over time from its own experiences. This could be achieved by comparing the effects of actions with resulting percepts. Cherniak (1986) introduced the minimal conditions of an agent to be rational. An autonomous agent determines its behaviour from its own experience. If the agent learns by interacting with the environment, performs actions and receives percepts via its own sensors it will build not only models which fit the agent's perception of the environment but also build a model in its own terms (Smith, 2000).

Thus, another direction for extending the research presented in this thesis would be to facilitate SLiDe within a framework of an autonomous agent-based designing system. SLiDe can be set within an agent-based designing system that interacts with an environment which is itself a model of the evolving design viewed as an external representation for the agent. Instead of the agent constructing elaborate internal world models within which to learn (as developed in SLiDe), the proposed agent would interact with the external representation imitating a designer. An initial framework is shown in Figure 8.2. The proposed framework constitutes the environment and the agent. The environment presents the external representation. The agent builds its view about the environment through its sensors. The agent has containers for information, called pools, and functions that process information and transfer it among pools. The external representation (EXT) contains a representation of the agent's environment. The pools contain representations of the agent's model. The sensors detect the environment, perceive its representation and place their output in the percepts (PERC) pool. Depending on the agent's goal and focus of attention, the external representation in the

EXT can be perceived in various ways. The perception function (P) determines what the agent perceives. Hence, it decides what sensors should be used to construct the percepts and place them in the (PERC) pool. The cognition function (C) provides a means of additional cognitive inferences based on the percepts in the (PERC) and constructs the current situation from its percepts and places its output in the situation pool (SIT). The handling function (H), given the current situation and goal situatedness, prepares some actions and places them in the actions pool (ACT). The action function (A) executes the actions prepared by the handling function (H) causing some transformation in the environment. In response to the new changes in the external representation the agent's sensors are activated and consequently the agent's updated view of the environment is constructed. Different views of the environment constructed by the agent allow for manipulating existing objects to discover new objects and relationships which are important activities in conceptual designing. The agent could discover such relationships from its interpretations of perceptions of the external representations. The agent's interpretations would be altered in response to the effects of its actions on the external representation. Since such an agent is assumed to useful in conceptual designing where the designing task proceeds without having sufficient knowledge at the outset to complete the design, then the agent needs not only to discover objects and relationships but also to recognise useful patterns so as to facilitate progress in designing.

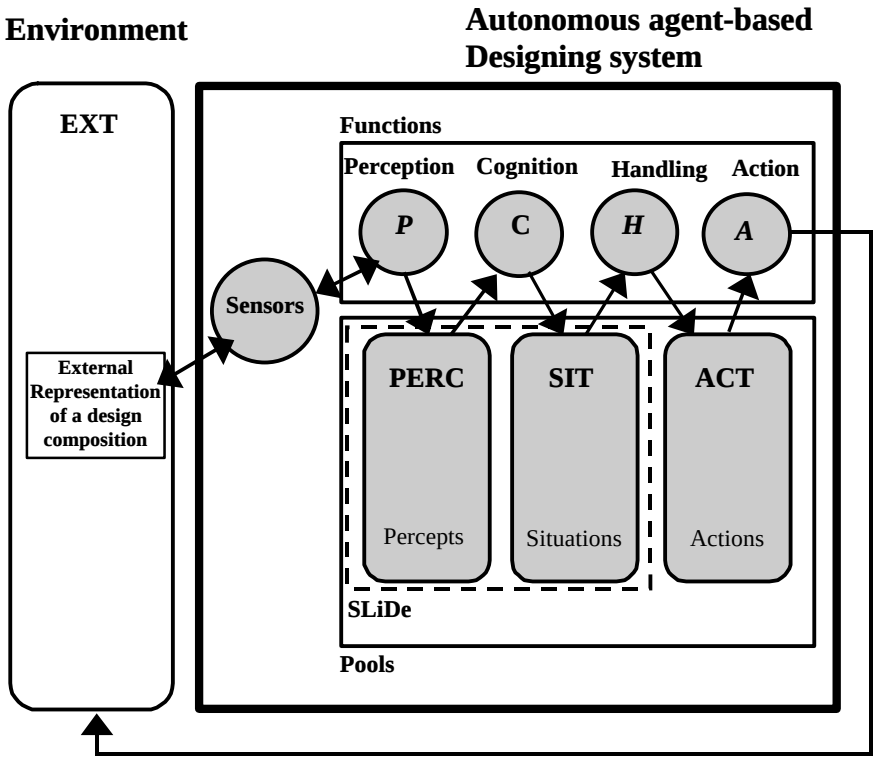


Figure 8.2 An initial framework of an autonomous agent-based designing system.

References

Agre, P. and Horswill, I. (1997). Lifeworld analysis, *Journal of Artificial Intelligence Research*, **6**: 111-145.

Agre, P. E. and Chapman, D. (1987). Pengi: An implementation of a theory of activity, *Proceedings of AAAI-87*, Morgan Kaufmann, Seattle, WA, pp. 278-272.

Ainsworth, S. E., Wood, D. J. and Bibby, P. A. (1996). Co-ordinating multiple representations in computer based learning environments, *in Proceedings of the European Conference on Artificial Intelligence and Education (EUROAI-ED)*, Lisbon.

Akin, Ö. (1986). *Psychology of Architectural Design*, Pion, London.

Akman, V. (1999). Situations and AI: Guest editor‘s introduction, *Minds and Machines*, **8**(4): 475-477.

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

Akman, V. and Surav, M. (1996). Steps toward formalizing context, *AI Magazine*, **17**(3): 55-72.

Archer, L. (1965). *Systematic Method for Designers*, Council of Industrial Design, London.

Arnheim, R. (1969). *Visual Thinking*, University of California Press, Berkeley.

Arnheim, R. (1977). *The Dynamics of Architectural Form: Based on the 1975 Mary Duke Biddle Lectures at the Cooper Union*, University of California Press, Berkeley.

Asimov, M. (1962). *Introduction to Design*, Prentice-Hall, Englewood Cliffs, NJ.

Babin, A. and Loganantharaj, R. (1991). Designer's workbench: A tool to assist in the capture and utilisation of design knowledge, *in* J. S. Gero (ed.), *Artificial Intelligence in Design '91*, Butterworth Heinemann, Oxford, pp. 249-267.

Baglivo, J. A. and Graver, J. E. (1983). *Incidence and Symmetry in Design and Architecture*, Cambridge University Press, Cambridge.

Baker, R. (1993). *Designing the Future: The Computer in Architecture and Design*, Thames and Hudson, New York.

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.

Bartlett, F. C. (1934). *Remembering: A Study in Experimental and Social Psychology*, The University Press, Cambridge, England (reprinted 1967).

Bartlett, F. C. (1958). *Thinking: An Experimental and Social Study*, Allen and Unwin, London.

Barwise, J. and Perry, J. (1983). *Situations and Attitudes*, MIT Press, Cambridge, Mass.

Barwise, J. and Seligman, J. (1997). *Information Flow: The Logic of Distributed Systems*, Cambridge University Press, Cambridge, UK.

Beer, R. (1990). *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology*, Academic Press, San Diego, CA.

Bereiter, C. (1997). Situated cognition and how to overcome it, *in* D. Kirshner and J. A. Whitson (eds), *Situated Cognition: Social, Semiotic, and Psychological Perspectives*, L. Erlbaum, Mahwah, N.J., pp. 281-300.

Beyer, D. M. (1998). From PuppyPaste to DogPaste: Toward a situated, learning analogy maker, *Master Thesis*, Department of Philosophy, Binghamton University, Binghamton, NY.

Bhatta, S. R. and Geol, A. K. (1994). Discovery of physical principles from design experiences, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, **8**: 113-123.

Billett, S. (1996). Situated learning: Bridging sociocultural and cognitive theorising, *Learning and Instruction*, **6**(3): 263-280.

Britt, B. D. and Glagowski, T. (1996). Reconstructive derivational analogy: A machine learning approach to automate design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, **10**: 115-126.

Brown, D. C. and Birmingham, W. P. (1997). Understanding the nature of design, *IEEE Expert*, **12**(2): 14-16.

Brown, J. S., Collins, A. and Duguid, P. (1989). Situated cognition and the culture of learning, *Educational Researcher*, **18**(1): 32-42.

Bruce, V., Green, P. R. and Georgeson, M. A. (1996). *Visual Perception: Physiology, Psychology, and Ecology*, Psychology Press, Hove, East Sussex.

Carbonell, J. G. (1990). *Machine Learning: Paradigms and Methods*, MIT Press, Cambridge, Mass.

Cha, M. Y. (1998). Architectural shape pattern representation and its implication for design computing, *Ph.D. Thesis*, Key Centre of Design Computing, Department of Architectural and Design Science, The University of Sydney, Sydney.

Cha, M. Y. and Gero, J. S. (1998). Shape pattern representation for design computation, *Working Paper*, Key Centre of Design Computing and Cognition, The University of Sydney, Sydney, Australia.

Chabot, R. and Brown, D. C. (1994). Knowledge compilation using constraint inheritance, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, **8**: 125-142.

Chase, S. (2000). User interaction in grammar based design systems: From interface analysis to formal models, *in Proceedings of Greenwich 2000: Digital Creativity Symposium*, Greenwich, England, pp. 61-70.

Chase, S. C. (1993). The use of multiple representations to facilitate design interpretation, *in Proceedings of ARECDAO '93*, Barcelona, Spain, pp. 205-217.

Chase, S. C. (1997). Logic based design modeling with shape algebras, *Automation in Construction*, **6**(4): 311-322.

Cheeseman, P. and Stutz, J. (1996). Bayesian Classification (AutoClass): Theory and Results, *in* U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy (eds), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo park, Calif, pp. 61-83.

Cherniak, C. (1986). *Minimal Rationality*, MIT Press, Cambridge, Mass.

Clancey, W. J. (1991). Situated cognition: Stepping out of representational flatland, *AI Communications-The European Journal on Artificial Intelligence*, **4**(2/3): 109-112.

Clancey, W. J. (1993). Situated action: A neuropsychological interpretation response to Vera and Simon, *Cognitive Science*, **17**: 87-116.

Clancey, W. J. (1997a). The conceptual nature of knowledge, situations, and activity, *in* P. Feltovich, R. Hoffman and K. Ford. (eds), *Human and Machine Expertise in Context*, The AAAI Press, Menlo Park, CA, pp. 247–291.

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

Clancey, W. J. (1998). Interactive coordination processes: How the brain accomplishes what we take for granted in computer languages, *in* Z. Pylyshyn (ed.), *Constraining Cognitive Theories: Issues and Options*, Ablex Publishing Corporation, Greenwich, pp. 165-190.

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

Coyne, R. D. and Gero, J. S. (1985). Design knowledge and context, *Environment and Planning B: Planning and Design*, **12**: 419-442.

Coyne, R. D., Rosenman, M. A., Radford, A. D., Balachandran, M. and Gero, J. S. (1990). *Knowledge-Based Design Systems*, Addison-Wesley, Reading, Mass.

Cross, N. (1999). Natural intelligence in design, *Design Studies*, **20**: 25-39.

Cross, N., Christiaans, H. and Dorst, K. (1996). *Analysing Design Activity*, Wiley, Chichester ; New York.

Damski, J. (1996). Logic representation of shapes, *PhD Thesis*, Department of Architectural and Design Science, The University of Sydney, Sydney, Australia.

Damski, J. and Gero, J. S. (1994a). A model of shape emergence based on human perception, *in Information Technology in Design, Vol. 2*, ICSTI, Moscow, pp. 96-105.

Damski, J. and Gero, J. S. (1994b). Visual reasoning as visual re-interpretation through re-representation, *in AID'94 Workshop on Reasoning with Shapes in Design*, Lausanne, pp. 16-20.

Das, S. K., Fox, J., Elsdon, D. and Hammond, P. (1997). A flexible architecture for autonomous agents, *Journal of Experimental & Theoretical Artificial Intelligence*, **9**(4): 407-440.

Dasgupta, S. (1989). The structure of design process, *Advances in Computers*, **28**: 1-67.

Davis, R., Shrobe, H. and Szolovits, P. (1993). What is knowledge representation?, *AI Magazine*, **14**(1): 17-33.

Dewey, J. (1939). *Logic: The Theory of Inquiry*, reprinted in 1982, Irvington, New York.

Do, E. Y.-L. (1998). The right tool at the right direction: Investigation of freehand drawings as an interface to knowledge based design tools, *Ph.D. Thesis*, College of Architecture, Georgia Institute of Technology, Atlanta, Georgia.

Dorst, K. and Dijkhuis, J. (1996). Comparing paradigms for describing design activity, *in* N. Cross, H. Christiaans and K. Dorst (eds), *Analysing Design Activity*, Wiley, Chichester, pp. 253-250.

Duffy, A. and Duffy, S. (1996a). Learning and design reuse, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, **10**(2): 139-142.

Duffy, A. H. B. (1997). The "What" and "How" of learning in design, *IEEE Expert*, **12**(3): 71-76.

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

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

Engestrom, Y. and Cole, M. (1997). Situated cognition in search of an agenda, *in* D. Kirshner and J. A. Whitson (eds), *Situated Cognition: Social, Semiotic, and Psychological Perspectives*, L. Erlbaum, Mahwah, N.J., pp. 301-309.

Errico, B. and Aiello, L. C. (1996). Intelligent agents in the situation calculus: An application to user modelling, *in* D. M. Gabbay and H. J. Ohlbach (eds), *Practical Reasoning: International Conference of Formal and Applied Practical Reasoning (FAPR'96),* Springer, Berlin, pp. 126-140.

Fisher, D., Xu, L., Carnes, J. R., Reich, Y., Fenves, S. J., Chen, J., Shiavi, R., Biswas, G. and Weinberg, J. (1993). Applying AI clustering to engineering tasks, *IEEE Expert,* **8**(6): 51-60.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering, *Machine Learning,* **2**: 130-172.

Fisher, D. H. and Schlimmer, J. C. (1988). Models of Incremental Concept Learning, *Technical Report 88-05*, Department of Computer Science, Vanderbilt University, Nashville, TN.

Galle, P. (1999). Design as intentional action: A conceptual analysis, *Design Studies,* **20**: 57-81.

Gedenryd, H. (1998). How designers work, *Ph.D Thesis*, Cognitive Science, Lund University, Lund, Sweden.

Gee, J. P. (1997). Thinking, learning and reading: The situated sociocultural mind, *in* D. Kirshner and J. A. Whitson (eds), *Situated Cognition: Social, Semiotic, and Psychological Perspectives*, L. Erlbaum, Mahwah, NJ, pp. 235-259.

Gennari, J. H., Langley, P. and Fisher, D. (1989). Models of incremental concept formation, *Artificial Intelligence,* **40**: 11-61.

Gero, J. S. (1990). Design prototypes: A knowledge representation schema for design, *AI Magazine,* **11**(4): 26-36.

Gero, J. S. (1992a). Creativity, emergence and evolution in design, *in* J. S. Gero and F. Sudweeks (eds), *Preprints Computational Models of Creative Design*, Department of Architectural and Design Science, University of Sydney, pp. 1-28.

Gero, J. S. (1992b). Shape emergence and symbolic reasoning using infinite maximal lines, *Unpublished Notes*, Design Computing Unit, Department of Architectural and Design Science, The University of Sydney, Sydney.

Gero, J. S. (1994). Towards a model of exploration in computer-aided design, *in* J. S. Gero and E. Tyugu (eds), *Formal Design Methods for CAD*, North Holland, Amsterdam, pp. 315-336.

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. (1998a). Conceptual designing as a sequence of situated acts, *in* I. Smith (ed.), *Artificial Intelligence in Structural Engineering*, Springer, Berlin, pp. 165-177.

Gero, J. S. (1998b). Situated learning in design, *in* A. Duffy (ed.), *AID'98 Workshop on Machine Learning in Design*, Lisbon, pp. 1-5.

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.

Gero, J. S., Damski, J. and Jun, H. (1995). Emergence in CAAD systems, *in* M. Tan and R. Teh (eds), *The Global Design Studio: Proceedings of the Sixth International Conference on Computer-aided Architectural Design Futures, CAAD Futures '95*, Centre for Advanced Studies in Architecture, National University of Singapore, Singapore, pp. 423-438.

Gero, J. S. and Fujii, H. (1999). A computational framework for concept formation for a situated design agent, *Working Paper*, Key Centre of Design Computing and Cognition, The University of Sydney, Sydney, Australia.

Gero, J. S. and Jun, H. (1995a). Getting computers to read the architectural semantics of drawings, *in* L. Kalisperis and B. Kolarevic (eds), *Computing in Design: Enabling, Capturing and Sharing Ideas*, ACADIA, pp. 97-112.

Gero, J. S. and Jun, H. (1995b). Visual semantic emergence to support creative design: A computational view, *in* J. S. Gero, M. L. Maher and F. Sudweeks (eds), *Preprints Computational Models of Creative Design*, University of Sydney, Sydney, Australia, pp. 87-117.

Gero, J. S. and Yan, M. (1993). Discovering emergent shapes using a data-driven symbolic model, *in* U. Flemming and S. V. Wyk (eds), *CAAD Futures'93*, Elsevier, Amsterdam, pp. 3-17.

Gero, J. S. and Yan, M. (1994). Shape emergence using symbolic reasoning, *Environment and Planning B: Planning and Design*, **21**: 191-218.

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.

Goel, V. (1995). *Sketches of Thought*, MIT Press, Cambridge, Mass.

Goldschmidt, G. (1991). The dialectics of sketching, *Design Studies*, **4**: 123-143.

Goldschmidt, G. (1994). On visual design thinking: the vis kids of architecture, *Design Studies*, **15**(2): 158-174.

Goldschmidt, G. (1997). Capturing indeterminism: Representation in the design problem space, *Design Studies*, **18**: 441-445.

Grecu, D. L. and Brown, D. C. (1996). Dimensions of learning in agent-based design, *in 4th International Conference on AI in Design - Workshop on Machine Learning in Design*, Stanford, CA, .

Grecu, D. L. and Brown, D. C. (1998). Guiding agent learning in design, *in* S. Finger, T. Tomiyama and M. Mantyla (eds), *Proc. of the 3rd IFIP Working Group 5.2 Workshop on Knowledge Intensive CAD*, Tokyo, Japan, pp. 237-250.

Greeno, J. G. (1989). Situations, mental models, and generative knowledge, *in* D. Klahr and K. Kotosvky (eds), *Complex Information Processing: The Impact of Herbert A. Simon*, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 285-318.

Greeno, J. G. (1995). Understanding concepts in activity, *in* C. A. Weaver, S. Mannes and C. R. Fletcher (eds), *Discourse Comprehension: Essays in Honor of Walter Kintsch*, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 65-95.

Heidegger, M. (1927). *Being and Time (translation, 1962)*, Harper & Row, New York.

Hert, C. A. (1997). Information retrieval as situated action, *Working Paper*, School of Library and Information Science, Indiana University, Bloomington, IN.

Hofmann, H. F., Pfeifer, R. and Vinkhuyzen, E. (1993). Situated software design, *in Proceedings of the Fifth International Conference on Software Engineering and Knowledge Engineering*, San Francisco, .

Huhns, M. N. and Acosta, R. D. (1992). An analogical reasoning system for solving design problems, *in* C. Tong and D. Sriram (eds), *Artificial Intelligence in Engineering Design*, Vol. 2, Academic Press, New York, pp. 105-143.

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.

Ivezic, N. and Garrett, J. H. (1994). A neural network-based machine learning approach for supporting synthesis, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, **8**: 143-161.

Jones, J. (1963). A method of systematic design, *in* Jones. J. and Thornley, D. (eds), *Conference on Design Methods*, Pergaman, Oxford, pp. 10-31; reprinted in 1984, *in* N. Cross (eds), *Developments in Design Methodology*, John Wiley, New York, pp. 9-31.

Jun, H. and Gero, J. S. (1998). Emergence of shape semantics of architectural shapes, *Environment and Planning B: Planning and Design*, **25**(4): 577-600.

Jun, H. J. (1997). Emergence of shape semantics of architectural drawings in CAAD systems, *Ph.D. Thesis*, Department of Architectural and Design Science, University of Sydney, Sydney.

Kalay, Y. E. (1989). *Modeling Objects and Environments*, Wiley, New York.

Karmiloff-Smith, A. (1993). Constraints on representational change: Evidence from children's drawing, *Cognition*, **34**: 57-83.

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. G. (1993). COBBIT - A control procedure for COBWEB in the presence of concept drift, *in Proceedings of the 1993 European Conference on Machine Learning*, Springer-Verlag, Vienna, pp. 244-261.

Kirsh, D. (1995). The intelligent use of a space, *Artificial Intelligence*, **73**: 31-68.

Kirshner, D. and Whitson, J. A. (1997). *Situated Cognition: Social, Semiotic, and Psychological Perspectives*, L. Erlbaum, Mahwah, NJ.

Kocabas, S. (1991). A review of learning, *The Knowledge Engineering Review*, **6**(3): 195-222.

Köhler, W. (1970). *Gestalt Psychology: An Introduction to New Concepts in Modern Psychology*, Liveright, New York.

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

Kolodner, J. L. (1993). *Case-Based Reasoning*, Morgan Kaufmann , San Mateo, CA.

Koutamanis, A. and Mitossi, V. (1993). Computer vision in architectural design, *Design Studies*, **14**(1): 40-57.

Landau, B. (1996). Multiple geometric representations of objects in languages and language learners, *in* P. Bloom (eds), *Language and Space*, MIT Press, Cambridge, Mass, pp. 317-363.

Langley, P. (1996). *Elements of Machine Learning*, Morgan Kaufmann, San Francisco.

Langley, P. (1999). Concrete and abstract models of category learning, *in Proceedings of the Twenty-First Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum, Vancouver, BC, pp. 288-293.

Lave, J. (1988). *Cognition in Practice: Mind, Mathematics, and Culture in Everyday Life*, Cambridge University Press, Cambridge, UK.

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

Lawson, B. (1980). *How Designers Think*, Architectural Press, London, UK.

Lenz, M. (1998). *Case-Based Reasoning Technology: From Foundations to Applications*, Springer, Berlin.

Li, H. (1994). *Machine Learning of Design Concepts*, Computational Mechanics, Southampton, UK.

Liu, T.-T. (1995). Some phenomena of seeing shapes in design, *Design Studies*, **16**(3): 367-385.

Logan, B. and Smithers, T. (1993). Creativity and design as exploration, *in* J. Gero and M. Maher (eds), *Modelling Creativity and Knowledge-Based Creative Design*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 139-175.

Lueg, C. (1997). An adaptive USENET interface supporting situated actions, *Proceedings of the 3rd ERCIM Workshop on User Interfaces for All*, Strasbourg, France, pp. 165-170.

Lueg, C. and Pfeifer, R. (1997). Cognition, situatedness, and situated design, *Proceedings of the Second International Conference on Cognitive Technology (CT 97)*, Aizu, Japan, pp. 124-135.

Lueg, C. P. (1999). Supporting situated information seeking: Communication, interaction and collaboration, *Ph.D. Thesis*, Department of Information Technology, University of Zurich, Zurich, Switzerland.

MacCallum, K. J., Duffy, A. H. B. and Green, S. (1987). An intelligent concept design assistant, *in* H. Yoshikawa and E. A. Warman (eds), *Design Theory for CAD*, IFIP W.G 5.2 Workshop 3 on Intelligent CAD, Elsevier Science, North-Holland, pp. 301-317.

Mackellar, B. and Peckham, J. (1992). Representing design objects in SORAC, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '92*, Kluwer Academic Publishers, Dordrecht, pp. 201-219.

Maes, P. (1990). Situated agents can have goals, *Robotics and Autonomous Systems*, **6**: 49-70.

Maher, M. L., Balachandran, M. and Zhang, D. M. (1995). *Case-Based Reasoning in Design*, Lawrence Erlbaum Associates, Mahwah, NJ.

March, L. and Steadman, P. (1971). *The Geometry of Environment: An Introduction to Spatial Organization in Design*, RIBA Pubs., London, UK.

Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W. H. Freeman, San Francisco.

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

McLaughlin, S. and Gero, J. S. (1987). Acquiring expert knowledge from characterized design, *Artificial Intelligence for Engineering Design, Analysis and Manufacture (AI EDAM)*, **1**(2): 73-87.

Meiss, P. v. (1991). *Elements of Architecture: From Form to Place*, Van Nostrand Reinhold, London, UK.

Michalski, R. S. and Stepp, R. (1983). Learning from observation: Conceptual clustering, *in* R. S. Michalski, J. Carbonell and T. Mitchell (eds), *Machine Learning: An Artificial Intelligence Approach*, TIOGA Publishing Co., Palo Alto, CA, pp. 331-363.

Mills, C. W. (1940). Situated actions and vocabularies of motive, *American Sociological Review*, **5**: 904-913.

Mitchell, T. M. (1982). Generalisation as search, *Artificial Intelligence*, **18**: 203-226.

Mitchell, T. M. (1997). *Machine Learning*, McGraw-Hill, New York.

Mitchell, W. J. (1990). *The Logic of Architecture: Design, Computation, and Cognition*, MIT Press, Cambridge, Mass.

Mitchell, W. J. and McCullough, M. (1995). *Digital Design Media*, Van Nostrand Reinhold, New York.

Morrison, C. T. (1998). Situated representation: Solving the handcoding problem with emergent structured representation, *Ph.D. Thesis*, Department of Philosophy, Binghamton University, New York.

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

Mostow, J., Barely, M. and Weinrich, T. (1992). Automated reuse of design plans in BOGART, *in* C. Tong and D. Sriram (eds), *Artificial Intelligence in Engineering Design*, Vol. 2, Academic Press, Boston, pp. 57-104.

Müller, J.-P. and Pecchiari, P. (1996). A model for systems of situated autonomous agents: An application to automated deduction, *ICMAS'96*, Kyoto, Japan.

Müller, M. and Pfeifer, R. (1997). Developing effective computer systems supporting knowledge intensive work: Situated design in a large paper mill, *in* M. Khosrowpour and J. Liebowitz (eds), *Cases on Information Technology Management in Modern Organization*, Idea Group Publishing, Hershey, PA, pp. 225-249.

Murdoch, T. and Ball, N. (1996). Machine learning in configuration design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, **10**: 101-113.

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.

Newell, A. (1982). The knowledge level, *Artificial Intelligence,* **18**(1): 87-127.

Nguyen, G. T. and Rieu, D. (1991). Representing design objects, *in* J.S. Gero (ed.), *Artificial Intelligence in Design '91*, Butterworth Heinemann, Oxford, pp. 367-386.

Nicolas Lachiche and Marquis, P. (1998). Scope classification: An instance-based learning algorithm with a rule-based characterisation, *in* C. Nédellec and C. Rouveirol (eds), *Machine Learning: ECML-98, 10th European Conference on Machine Learning, Lecture Notes in Computer Science, Vol. 1398*, Springer, Berlin, pp. 268-279.

Oehlmann, R., Edwards, P. and Sleeman, D. (1995). Self-questioning and experimentation: an index vocabulary of situated interaction, *in* I. Watson (ed.), *Progress in Cased -Based Reasoning*, Springer-Verlag, Berlin, pp. 59-72.

Ores, B. V. (1998). From context to contextualising, *Learning and Instruction,* **8**(6): 473-488.

Oxman, R. (1997). Design by re-representation: A model of visual reasoning in design, *Design Studies,* **18**: 329-347.

Oxman, R. M. (1995). The reflective eye: Visual reasoning in design, *in* A. Koutamanis, H. Timmerman and I. Vermeulen (eds), *Visual Data Bases in Architecture*, Averbury, Aldershot, UK, pp. 89-112.

Persidis, A. and Duffy, A. (1991). Learning in engineering design, *in* H. Yoshikawa, F. Arbab and T. Tomiyama (eds), *Intelligent CAD III*, Elsevier Science Publishers, Amsterdam, pp. 251-272.

Pfeifer, R. and Rademarkers, P. (1991). Situated adaptive design: Towards a methodology for knowledge systems development, *Proceeding of the Conference on Distributed Artificial Intelligence and Cooperative Work*, Springer-Verlag, Berlin, pp. 53-64.

Presidis, A. and Duffy, A. (1991). Learning in engineering design, *in* H. Yoshikawa, F. Arbab and T. Tomiyama (eds), *Intelligent CAD III*, Elsevier Science, pp. 251-272.

Purcell, T. and Gero, J. S. (1998). Drawings and the design process, *Design Studies,* **19**: 389-430.

Quinlan, J. R. (1986). Induction of decision trees, *Machine Learning,* **1**: 81-106.

Rademakers, P. and Pfeifer, R. (1992). The role of knowledge level in situated design, *AI Memo '92*, Artificial Intelligence Laboratory, Vrije Universiteit Brussel, Brussels, Belgium.

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

Rappaport, A. T. (1998). Constructive cognition in a situated background, *International Journal of Human Computer Studies,* **49**(6): 927-933.

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.

Reich, Y. (1994). Towards practical machine learning techniques, *in Proceedings of the First Congress on Computing in Civil Engineering (Washington, DC)*, ASCE, New York, NY, pp. 885-892.

Reich, Y. (1997). Machine learning techniques for civil engineering problems, *Microcomputers in Civil Engineering,* **12**(4): 295-310.

Reich, Y. (1998). Learning in design: From characterizing dimensions to working systems, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, **12**(2): 161-172.

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., Konda, S., Levy, S. N., Monarch, I. and Subrahmanian, E. (1993). New roles for machine learning in design, *Artificial Intelligence in Engineering,* **8**(3): 165-181.

Rendell, L. (1988). Learning hard concepts through constructive induction: Framework and rationale, *Technical Report UIUUCDCS-R-88-1426*, Department of Computer Science, University of Illinois as Urbana-Champagin, Urbana.

Robbins, E. and Cullinan, E. (1994). *Why Architects Draw*, MIT Press, Cambridge, Mass.

Rosenman, M. A. and Gero, J. S. (1998). Purpose and function in design, *Design Studies,* **19**(2): 161-186.

Rosenschein, S. J. and Kaelbling, L. P. (1995). A situated view of representation and control, *Artificial Intelligence,* **73**(1-2): 149-173.

Rossignac, J. R., Borrel, P. and Nackman, L. (1989). Interactive design with sequences of parameterised transformations, *in* V. Akman, P. J. Hagen and P. J. Veerkamp (eds), *Intelligent CAD Systems II: Implementational Issues*, Springer-Verlag, Berlin, pp. 92-125.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Upper Saddle River, NJ.

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

Schön, D. A. (1983). *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York.

Schön, D. A. (1987). *Educating the Reflective Practitioner*, Jossey-Bass, San Francisco.

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

Sim, S. K. and Duffy, A. H. B. (1998). A foundation for machine learning in design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM),* **12**(2): 193-209.

Simon, H. A. (1996). *The Sciences of the Artificial*, MIT Press, Cambridge, Mass.

Smith, B. (1999). Situatedness/embeddedness, *in* F. C. Keil and R. A. Wilson (eds), *The MIT Encyclopedia of the Cognitive Sciences*, MIT Press, Cambridge, Mass, .

Smith, G. (2000). An argument for external representations in conceptual design, *Unpublished Report*, Key Centre of Design Computing and Cognition, The University of Sydney, Sydney, Australia.

Smithers, T., Tang, M., P., R. and Tomes, N. (1993). Supporting drug design using an incremental learning approach, *Artificial Intelligence in Engineering,* **8**(3): 201-216.

Solso, R. L. (1997). *Cognition and the Visual Arts*, MIT Press, Cambridge, Mass.

Sooriamurthi, R. and Leake, D. (1994). Towards situated explanation, *in Proceedings of the 12th National conference on Artificial Intelligence*, AAAI Press, Cambridge, MA, pp. 1492.

Soufi, B. and Edmonds, E. (1996). The cognitive basis of emergence: Implications for design support, *Design Studies*, **17**: 451-463.

Stahl, G. (1993). Supporting situated interpretation, *Proceedings of the Cognitive Science Society: A Multidisciplinary Conference on Cognition*, Boulder, pp. 965-970.

Stiny, G. (1980). Introduction to shape and shape grammars, *Environment and Planning B: Planning and Design*, **7**: 343-351.

Stiny, G. (1990a). What designers do that computers should, *in* W. J. Mitchell, M. McCullough and P. Purcell (eds), *The Electronic Design Studio: Architectural Knowledge and Media in the Computer Era*, MIT Press, Cambridge, Mass, pp. 17-30.

Stiny, G. (1990b). What is a design?, *Environment and Planning B: Planning and Design*, **17**: 97-103.

Streibel, M. J. (1995). Instructional plans and situated learning, *in* G. J. Anglin (ed.), *Instructional Technology: Past, Present, and Future*, Libraries Unlimited, Englewood, Colo., pp. 145-160.

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

Sun, D. (1993). Memory, design and the role of computers, *Environment and Planning B: Planning and Design*, **20**: 125-143.

Suwa, M., Gero, J. S. and Purcell, T. (1998a). The roles of sketches in early conceptual design processes, *in Proceedings of Twentieth Annual Meeting of the Cognitive Science Society*, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 1043-1048.

Suwa, M., Purcell, T. and Gero, J. S. (1998b). Macroscopic analysis of design processes based on a scheme for coding designers' cognitive actions, *Design Studies*, **19**(4): 455-483.

Suwa, M., Gero, J. S. and Purcell, T. (1999). Unexpected discoveries and s-inventions of design requirements: A key to creative designs, *in* J. S. Gero and M. L. Maher (eds), *Computational Models of Creative Design IV*, Key Centre of Design Computing and Cognition, University of Sydney, Sydney, Australia, pp. 297-320.

Tan, M. (1990). Saying what it is by what it is like: Describing shapes using line relationships, *in* W. J. Mitchell, M. McCullough and P. Purcell (eds), *The Electronic Design Studio: Architectural Knowledge and Media in the Computer Era*, MIT Press, Cambridge, Mass, pp. 201-213.

Thelen, E. and Smith, L. B. (1994). *A Dynamics Systems Approach to the Development of Cognition and Action*, MIT Press, Cambridge, Mass.

Tong, C. (1992). Using exploratory design to cope with design problems complexity, *in* C. Tong and D. Sriram (eds), *Artificial Intelligence in Engineering Design*, Vol. 2, Academic Press, New York, pp. 287-332.

Turney, P. (1996). The identification of context-sensitive features: A formal definition of context for concept learning, *in ICML-96 Workshop on Learning in Context-Sensitive Domains, at the 13th International Conference on Machine Learning*, Bari, Italy, pp. 53-59.

Valkenburg, R. and Dorst, K. (1998). The reflective practice of design teams, *Design Studies*, **19**: 249-271.

Vera, A. H. and Simon, H. A. (1993). Situated action: A symbolic interpretation, *Cognitive Science*, **17**: 7-48.

Wang, J. and Howard, H. C. (1994). Recording and reuse of design strategies in an integrated case-based design systems, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, **2**: 219-238.

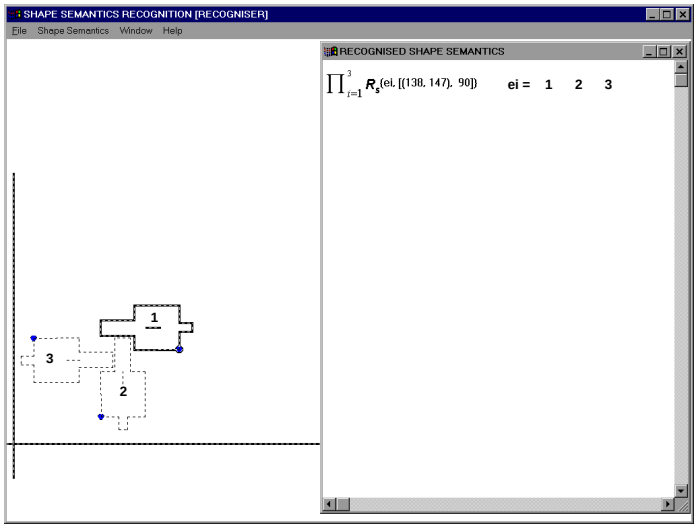
Wheeler, M. (1994). For whom the bell tolls? the roles of representation and computation in the study of situated agents, *Working Paper*, School of Cognitive and Computing Science, Brighton, UK.

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

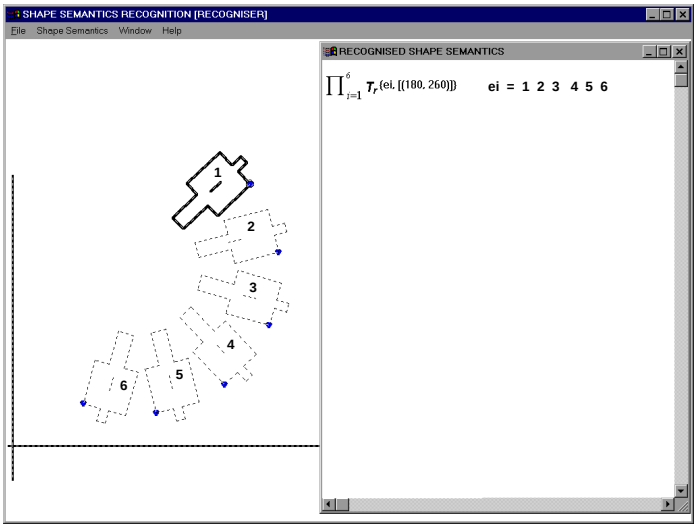
Appendix A

Recognising various Shape Semantics using the Recogniser module of SLiDe

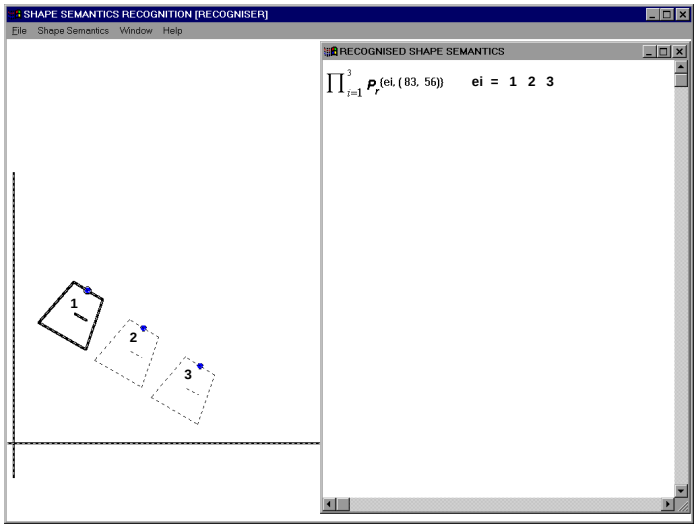
Recognition of Simple Rotation (R_s)



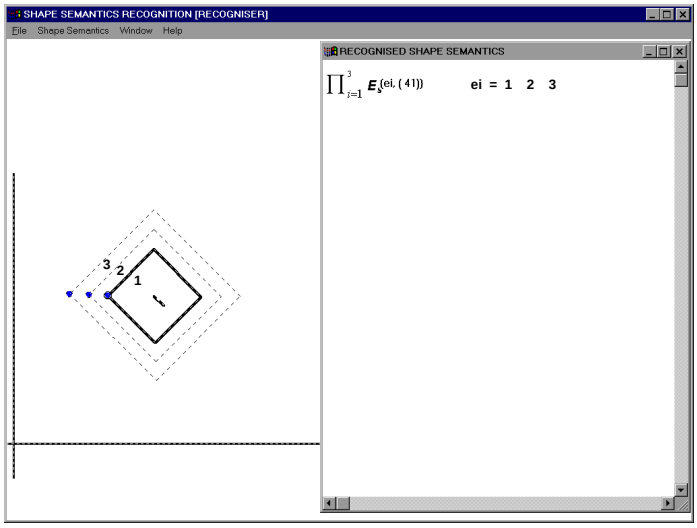
Recognition of Radiality (T_r)



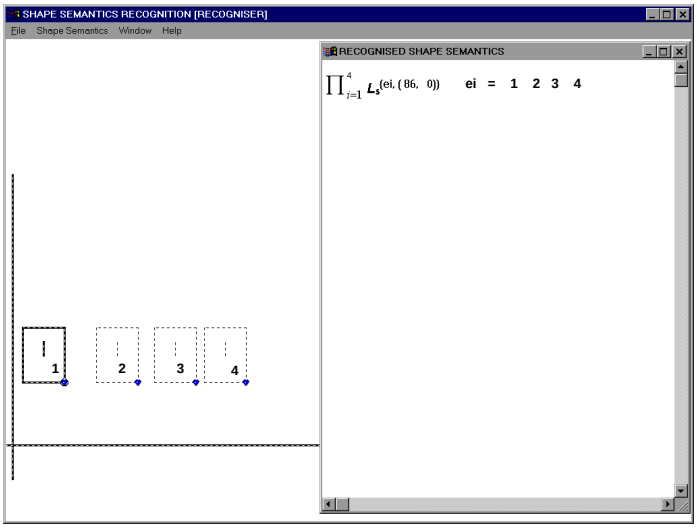
Recognition of Translational Repetition (P_r)



Recognition of Scaling (E_s)

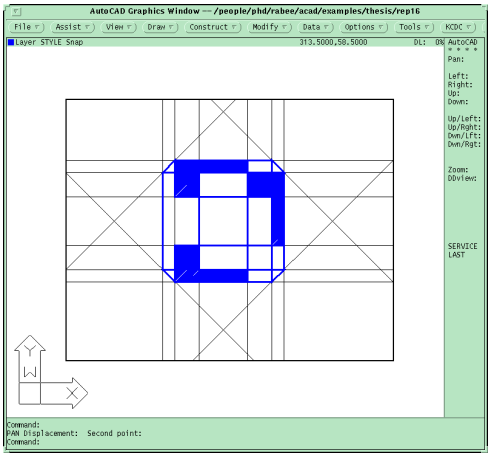


Recognition of Linearity (L_s)

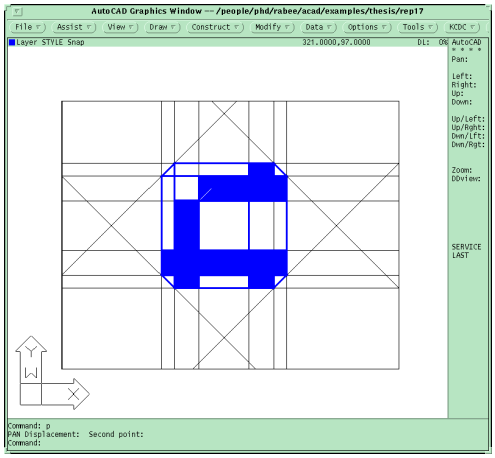


Appendix B

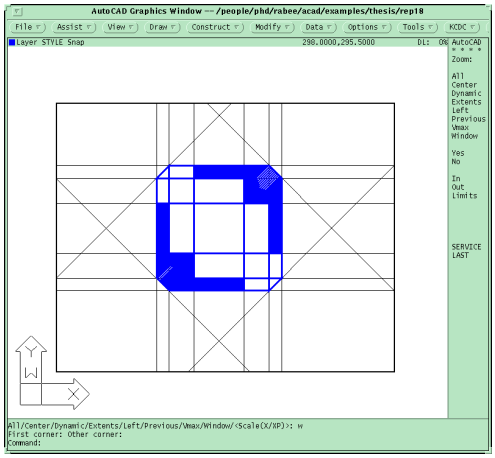
The fourth set of representations developed using the Generator module of SLiDe



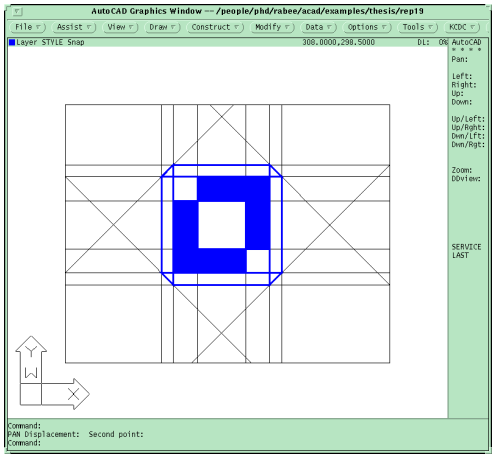
N_{16}



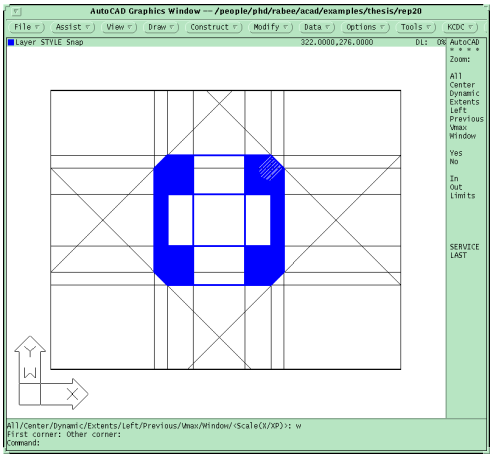
N_{17}



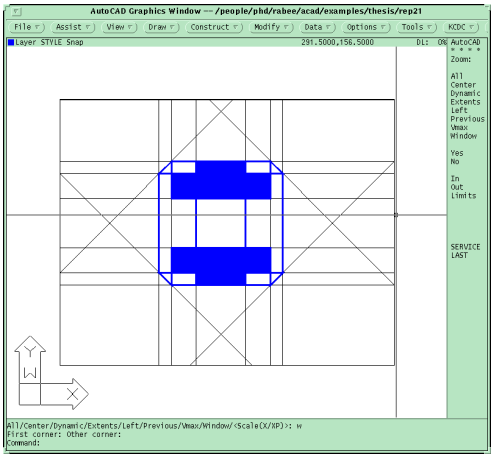
N_{18}



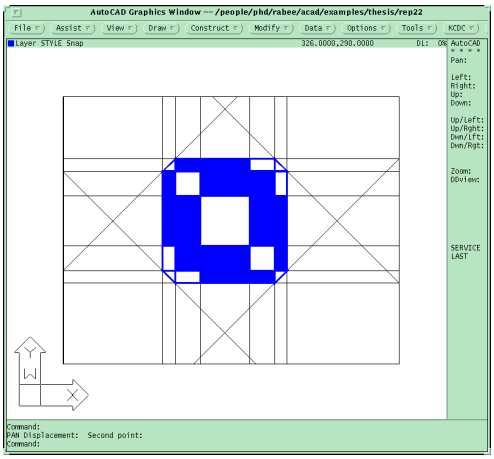
N_{19}



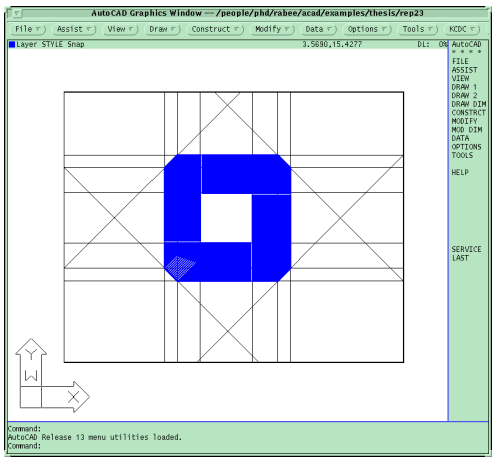
N_{20}



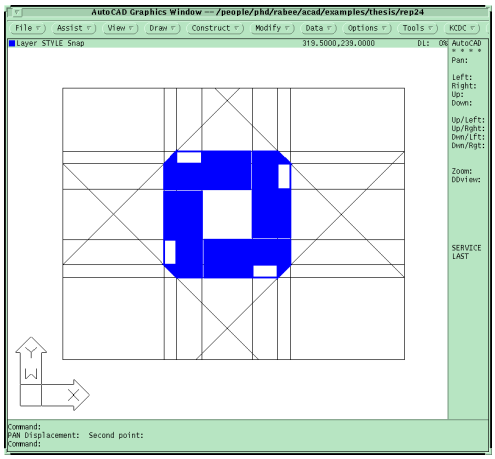
N_{21}



N_{22}



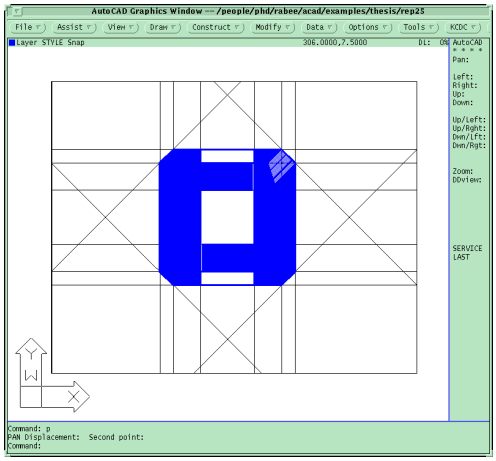
N_{23}



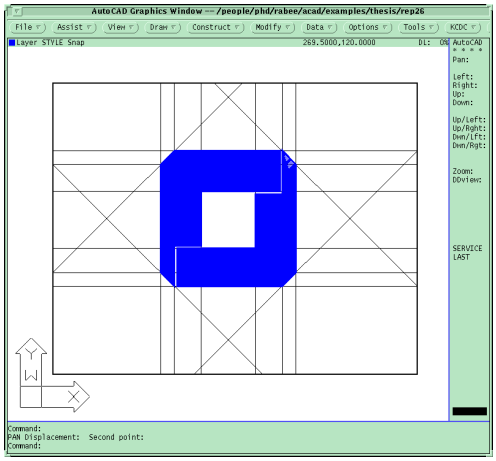
N_{24}

Appendix C

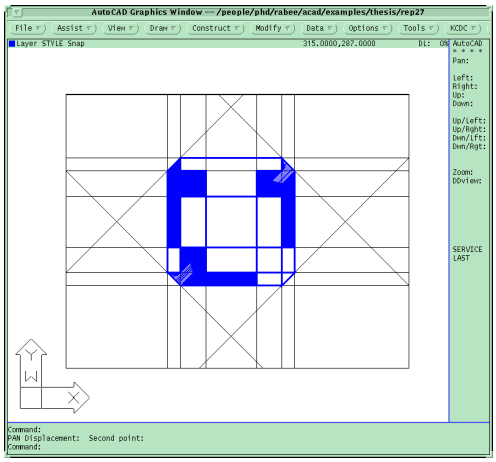
The fifth set of representations developed using the Generator module of SLiDe



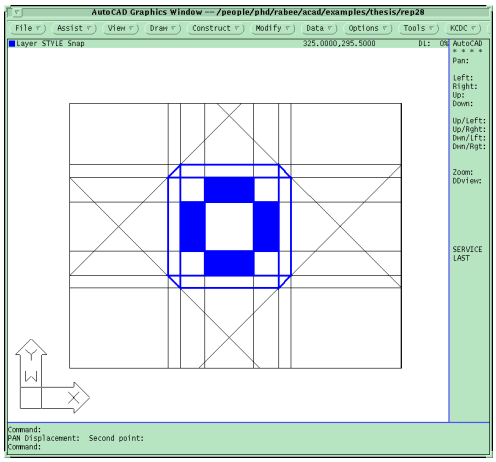
N_{25}



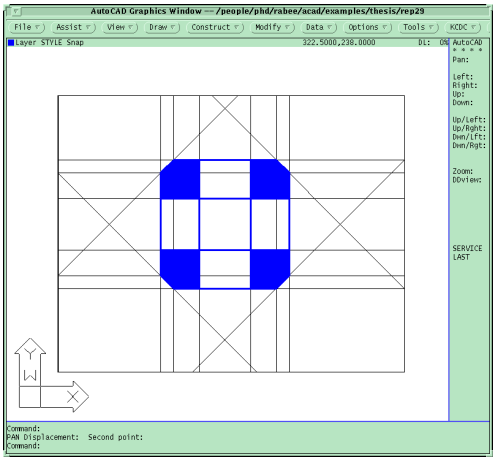
N_{26}



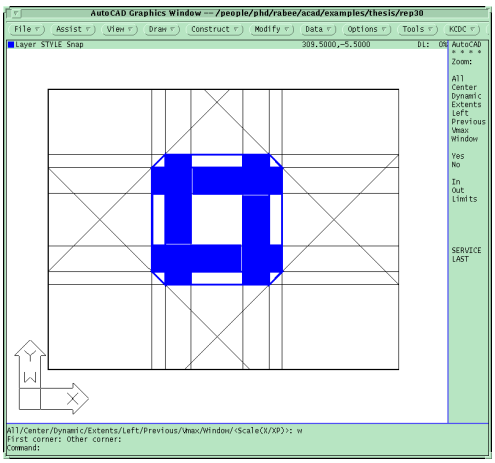
N_{27}



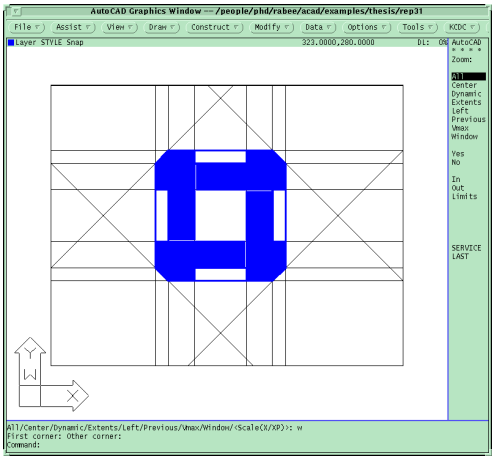
N_{28}



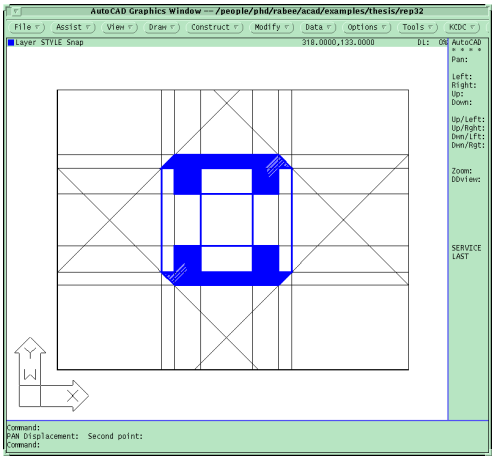
N_{29}



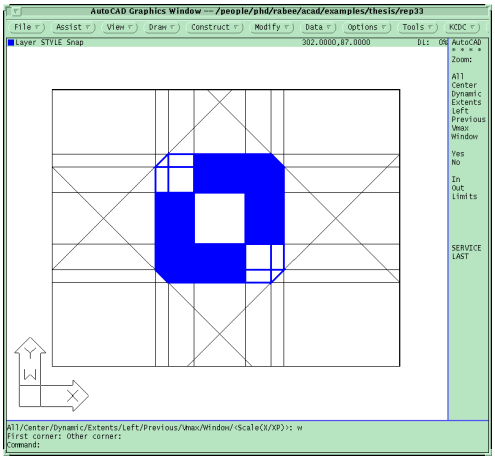
N_{30}



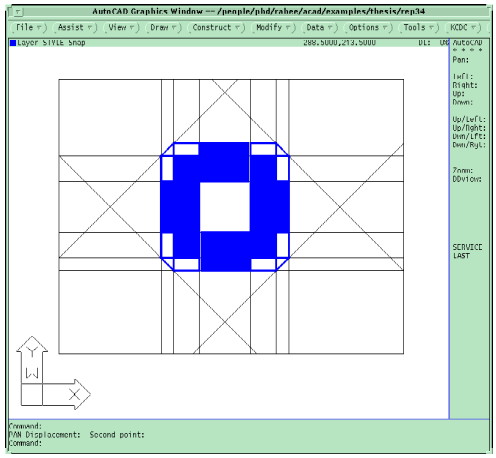
N_{31}



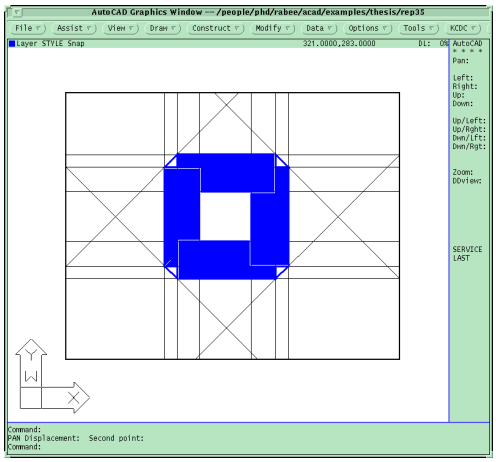
N_{32}



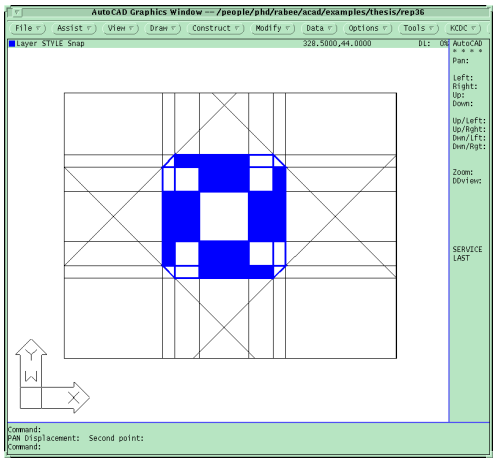
N_{33}



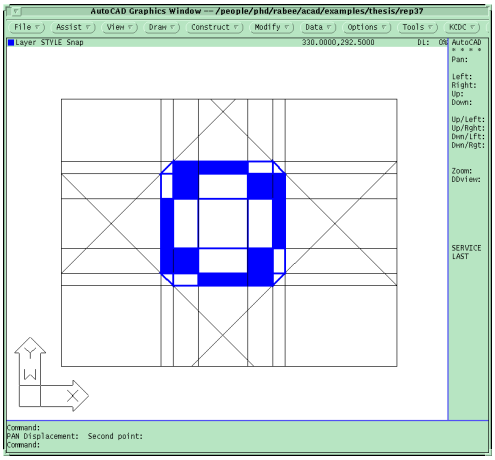
N_{34}



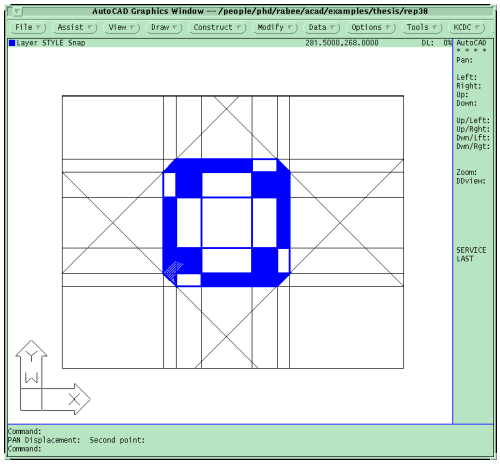
N_{35}



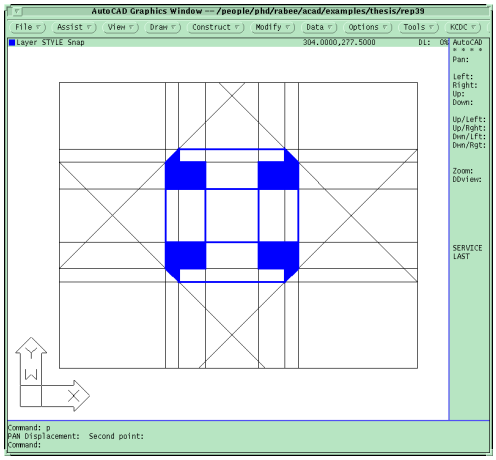
N_{36}



N_{37}



N_{38}



N_{39}

Appendix D

Abstracts of Papers Published from the Research in this Thesis

(1) Gero, J. S. and Reffat, R. M. (to appear). Multiple representations as a platform for situated learning systems in designing, *Knowledge Based Systems*.

This paper introduced the development of multiple representations as a platform for learning design knowledge in relation to the situations within which it was recognised. The benefits of this approach derive from the fact that knowledge is more useful when it is learned in relation to its situation and less useful when it is learned out of context. The situation is the way in which knowledge is located in relation to its surroundings. The situatedness of knowledge is constructed through learning which parts of the surroundings are in conjunction with it across different representations of a design composition. In order to learn the situatedness of design knowledge a medium is needed to present the design composition from different views, each of which allows for various situations to be encountered. What makes multiple representations useful in the context of situatedness is that they provide the opportunities for different and rich relationships among design knowledge to be constructed. This provides a system within which to learn from a number of representations in which the situatedness of knowledge can be discerned and learned. Architectural design compositions were chosen as a vehicle for the demonstration of the concept of situatedness in designing because the discovery of relationships among parts of the design composition are fundamental tasks in designing. The paper showed how multiple representations could provide a platform for situated learning systems in designing. What kind of situated knowledge could be learned from some of the possible representations of an architectural design composition is discussed. The regularities of relationships between design knowledge and its situations are investigated.

(2) Reffat, R. M. and Gero, J. S. (2000). Computational situated learning in designing: Application to architectural shape semantics, *in* J. S. Gero (ed.), *Artificial Intelligence in Design'00*, Kluwer, Dordrecht (to appear).

This paper presented the development of a computational system of Situated Learning in Designing (SLiDe). Situated learning is based on the concept 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 its situation within which it was recognised. SLiDe elucidates how design

knowledge is learned in relation to its situation and how designing 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 concept drift), that makes it capable of constructing various situational categories and modifying them over time. The paper concluded with a discussion of the potential benefits of using SLiDe during the conceptual stages of designing.

(3) Reffat, R. M. and Gero, J. S (2000). Towards active support systems for architectural designing, *Education of Computer Aided Architectural Design in Europe, eCAADe18 2000* (to appear).

This paper proposed the application of a situated learning approach in designing integrated with a conventional CAD system. The approach is implemented in SLiDe (Situated Learning in Designing) and integrated as SLiDe-CAAD, to provide interactive support in designing exemplified within the composition of architectural shapes. SLiDe-CAAD is proposed to assist in exploring the design space for various alternatives of design compositions; recognising shape semantics from a design composition; and in maintaining the integrity of shape semantics or desired design concepts of interest in the design composition. SLiDe-CAAD is introduced to provide a collaboration between the designer and the computer during the process of designing.

(4) 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.

In this paper we adopted the approach that designing is a series of situated acts, ie designing cannot be pre-planned to completion. This is based on ideas from situated cognition theory that claims that what people perceive, how they conceive and what they do develop together and are adapted to the environment. For a system to be useful for human designers it must have the ability to associate what is learned to its environment. In order for a system to do that such a system must be able to acquire knowledge of the environment that a design constructs. Therefore, acknowledging the concept of situatedness is of importance to provide a system with such capability and add on a new dimension to existing learning systems in design. A situated learning system in designing (SLiDe) is developed and has the capability to construct its own situational categories from its perceptual experiences and modify them if encountered again to link the learned knowledge to its corresponding situation. We have chosen architectural shapes as the vehicle to demonstrate our ideas and used multiple representations to build a platform for a SLiDe to learn from. In this paper the concept of situatedness and its role in both designing and learning are discussed. The framework of a SLiDe is introduced and how the potential

outcome of such a system will support human designers while designing is discussed.

(5) Reffat, R. M. and Gero, J. S. (1998). Learning about shape semantics: A situated learning approach, *in* T. Sasada, S. Yamaguchi, M. Morozumi, A. Kaga and R. Homma (eds), *Proceedings of CAADRIA '98*, CAADRIA, Kumomoto, Japan, pp. 375-384.

Designers recognise or make sense of objects in the context "situations" of other things. Designing cannot be predicted. The designer has to be "at a particular set of states" in order to decide what to do. The inability to determine a priori all design states implies that any design process cannot be pre-planned and design actions cannot be pre-defined. Situated learning is based on the concept that knowledge is contextually situated and is fundamentally influenced by the context in which it is used. We proposed a situated learning approach in the domain of architectural shapes designing. This paper elaborated the concept of situated learning and demonstrated what it produced in the domain of shape semantics.

(6) Gero, J. S. and Reffat, R. M. (1997). Multiple representations for situated agent based learning, *in* B. Verma and X. Yao (eds), *ICCIMA'97*, Griffith University, Gold Coast, Queensland, Australia, pp. 81-95.

Designers interact with the world not as actors following preconceived plans but relate to the situations encountered. Learning the situatedness of design knowledge is as important as learning design knowledge. Knowledge can be represented in many ways. Multiple representations combine the advantages of different representational forms within one system. Situated agent-based learning discovers and acquires useful knowledge and recognises the situation from multiple representations of the knowledge. In this paper, a model of a situated agent-based learning is described and the use of multiple representations in learning knowledge is presented.