Case Based Reasoning For Design Composition In Architecture

Ph.D. Thesis Proposal

Kamal Mubarak
School of Architecture
Carnegie Mellon University

Advising Committee

Prof. Susan Finger [chair] - Civil Engineering Dept.
Prof. Ulrich Flemming - School of Architecture
J. Secosky - R.A. Facilities Management Dept.
All from Carnegie Mellon University

FALL 2003

Abstract

This thesis proposal presents a process model for design composition using a representation for the formal characteristics of design solution composition. The design process model is based on adapting previous solutions to generate new designs using derivational analogy. The model relies on a visual representation of design compositions. This representation encodes functional and form attributes of design and is used to build compositional characteristics of the design. The research proposes the use of Solution Traces (Sol-Traces) as constructs for recording and reusing the design composition path which including history, alternatives, and decisions.

Design reasoning with architectural cases has a long history in architecture. In Case Based Reasoning (CBR), an established methodology in Artificial Intelligence, the results of previous cases are applied to new situations, thus reducing the complexity of necessary reasoning and allowing a problem solver to anticipate and avoid previous mistakes. Generative CBR using derivational analogy is a powerful problem solving technique which enables new designs to be created by utilizing the generative path of prior designs. This technique is being adapted for use in developing a design assistance system for design composition in architecture. Sol-Traces are used within a CBR methodology in a computer system for design composition assistance. The research proposes a new technique for computer assisted design composition. A prototype system utilizing this approach is proposed as a proof of concept. The envisioned system provides implementation for crucial aspects of the approach: the Sol-Traces formalism and the process of design composition.

Table of Contents

1. Introduction

The goal of this research is to provide computer assistance for designers in the early phases of the design process when they are composing design solutions. The research relies on the notion that designers remember previous design episodes and use them in composing new designs. Consider the following episode:

> …A designer is composing a new form for a building. He browses his collection of similar buildings designed by well-known designers. He tries to find something new, something creative. He admires the work of one designer but he does not want to imitate it. Browsing, viewing, browsing again; he is spreading his search widely. He sketches an initial solution. He explores deeper into the examples and designs. He focuses on one particular design; he abstracts; and he analyses it some more. Suddenly, he focuses on an emerging solution. He sketches more and is surprised with the result; a concept is emerging from that particular design, he guides the emergence in the way to fit the new requirements. He is pleased with the result and keeps developing the concept …

A new design is a creative solution, which emerges only when the designer traces a prior concept to its origin, working through its derivation path. Variations of this episode are common in the design world. The collection of prior designs could be the designer's own buildings or a wide range of design precedents. The common element in these episodes is the tracing back, the reaching for design concepts, and the emergence of a creative derivation.

Providing computer assistance for tracing and transforming prior designs is the subject of this research. This derivation may be referred to as *design rationale*, *design intent*, or *design history*. In this research, we will use the term solution trace, or *Sol-Trace* to denote the prior design and the process by which it is transformed into a new design. The research focuses on computational methods for representing these traces, storing them, retrieving them, and adapting them to create new solutions.

1.1 Building industry and the need for computational support

In architecture, early design is mainly done manually, not on computers, which results in a particular set of problems in practice. Schedules are not always met; previous design mistakes are repeated; and many times designs have to be redone or undergo radical changes which result in cost and time overruns. Assuring quality during the design phase is preferable to during the construction phase, when substantially larger cost is incurred for corrections. Frequently, design management is faced with trade offs between design quality and process time. There is a need to enhance this process through feedback and through other methods and techniques. We need to reduce the gap between the design phases, particularly between the early phases, and the other phases of building process.

Computation has promising potential in enhancing creative thinking. This already has been realized in many artistic fields such as graphic arts and musical compositions [Novah, 2001; Bruton, 1977]. Therefore, there is a compelling need to utilize computers in the conceptual composition of architectural design. The proposed research targets the way the design assistance can be achieved in the early phases in order to facilitate the process, predict and avoid failures, keep track of success, and develop innovative

and creative solutions. Providing such assistance can also help in insuring the quality of the design even under tight schedules. Eventually this will improve the efficiency in the building industry in general.

1.2 Formal methods and Computer aided design (CAD)

The term computer aided design (CAD) refers to a wide range of computational assistance to the design process. It ranges form simple or marginal assistance to complete design automation. In architecture, this assistance started with developing formal methods for describing the design process and products. Design patterns, as suggested by Christopher Alexander, were among the first attempts to formalize the design process [Alexander, 1977]. The need to manage and formalize the process continued in the design methods movement of the early sixties when pioneers such as Christopher Jones started to describe the design process in the form of methods [Jones, 1963]. This movement did not achieve fruitful results and design could not be confined merely within methods, but it did bring attention to the field of design research. Following the methods era, computers and computer aided design was introduced in the late seventies. Expectations were high and pioneers dreamed of a machine that could compose a design solution [Mitchell et al., 1977].

During the past two decades, the design process has been represented and described in several ways to make such dream to be realized. Design problems have been reduced to a set of small or abstracted processes that can be represented, performed, and managed computationally. Yet the dream has not been realized and the results are still far from satisfying. In the late eighties and early nineties, the CAD research community explored the cognitive aspects of the design process through protocol analysis, and other techniques [Cross et al., 1994].

Now we know more about the design process than two decades ago, only to realize that the design process can only be computationally assisted rather than automated. Computers can perform some tasks well and exceed human capabilities; these tasks include remembering and numeric calculations. At the same time, only humans can perform well and, at lest to date, exceed the capabilities of computers on tasks such as complex reasoning and design. With this understanding, many systems have been proposed to assist designers. These systems target specific components of the design process, such as functional analysis, shape manipulations, and visualization. With advances in computational hardware and software, CAD techniques are being re-examined and re-tested.

The goal of this research is to provide assistance in the early phases of the design process. The research builds on the fact that designers use their education and expertise in the field when designing new solutions. This thesis utilizes a new approach, in design knowledge acquisition and in developing knowledge based systems for building design, in order to overcome many difficulties associated with the rule-based approaches. The research explores questions such as; can the machine mimic designer's behavior in recalling expertise? Can the representation of expertise knowledge be a reliable source for solutions superior to designer's memory? Can this memory assist in solving design problems and composing new design solutions? The research uses a Case Based Reasoning (CBR) methodology to address these questions.

1.3 Case based reasoning (CBR)

This research uses CBR, both as a memory organization and as a problem solving methodology. CBR uses a knowledge representation technique called 'cases'. Through recalling these cases and reasoning with them, solutions to similar situations can be found. CBR systems can be viewed as a form of knowledge based systems or expert systems except that CBR does not consist of generalized rules but a memory of stored cases recording specific episodes, situations, or lessons. CBR techniques have been used successfully in many applications in different domains

The proposed research adapts CBR to suit the reasoning process in design: the solution process is shared between computers and designers; knowledge is partially represented; and designers provide the required 'glue' to assemble this knowledge in a new design solution. This research will examine CBR as a problem solving technique and apply it to one of the most demanding tasks; design composition in architecture. Also, CBR will be used as a computational memory of design cases.

In the design practice, previous experience or previous design solutions are represented, stored, retrieved, and reused in many ways. This experience sometimes referred to as design precedents or design memory [Oxman, 1994]. In fact, the design process can be viewed as a stream of episodes of previous situations reused and adjusted to produce a coherent new design solution. As quoted from Bermudez by Downing, design memory plays a large role in many aspects of the design process:

> Memory is knowledge. Memory can be used to help simulate new situations by means of exemplars. Memory liberates and traps: it librates by giving framework for action, from which such action then may depart; it traps, as the framework may be so strong that it proves impossible to break free of it. Memory is, to an extent, the universe of discourse. It establishes the reference point from which criteria and exemplars are utilized to accept/reject/develop ideas. Memory structures and shapes our perception of reality. Memory is generated and generates (supports and supported by) a set of social habits used in everyday life to ensure social interaction (predictability). A good design use of memory should avoid its direct, literal utilization; otherwise, the mind avoids inquiry and falls into mechanical, uncritical behavior and stereotypes. The problematization of memory is essential. One does it by de-framing the context of thought, so that memory has to be used in different, indirect manner. Design, as a process of action/inquiry, develops a memory.
>
> Julio Bermudez
>
> Quoted by Frances Downing in *Remembrance and the Design of Place* [Downing, 2000]

This thesis builds on the notion of design memory or expertise and develops a methodology for utilizing this memory in finding and composing design solutions. The CBR approach is based on remembering and encoding of previous design expertise. Using CBR as the main methodology in a computer assistance system provides a test of this concept of utilizing memory in design. From a computational point of view, CBR has the potential of providing innovative and creative design compositions.

1.4 Overview of proposal

This research proposes an abstraction for architectural design composition, at the early phases of the design process. The abstraction is represented through a visual language which is computationally represented while visually presented to the designers. Cases in this research are previous architectural

designs, represented in 2D floor plans. Solution Traces constructs are proposed to hold the compositional characteristics (path) of the design, for both function and form. The research proposes a toolkit for creating and adapting these traces. New designs are developed through the reuse, adaptation, and replay of Solution Traces.

Chapter 2 covers the required background and literature review for this approach of supporting design composition. A detailed description of the research problem, motivation, and objectives are provided in Chapter 3. In Chapter 4, the abstraction, the visual representation, and Solution Traces are explained, along with the used CBR methodology. This followed by implementation issues in Chapter 5. The expected tasks and schedule of the research are in Chapter 6. Finally, conclusions and expected contributions are in chapter 7.

2. Background and literature review

This chapter includes background about design composition and computer support for design composition, analogical reasoning in architectural design, and case based reasoning. The use of typology in design is also reviewed for the application of CBR in solving design problems. The state of the art of CBR applications in design composition is reviewed, including approaches, systems, and comments on opportunities for advancing the field.

2.1 Design composition

Design composition can be defined as the assembling of design components in a way that gives a specific character to the solution. Design composition in architecture is often referred to as the 'order' in the design solution. This order is introduced during the design process and can be seen in the floor plan of the building. The design order goes beyond the visual order, as it is connected to many other aspects such as components types and forms, functional relations between components, and construction techniques. In many cases during the design process, introducing order helps generate a better solution that fulfills the requirements.

Design composition, also called form design, is the process of imparting this order to the building form. Buildings are mostly judged by these forms, and reaching the best creative form is always a rewarding endeavor. Design theorists have been interested in describing order in design and how to compose a design form that satisfies a wide range of needs. Principles of form design are well presented in literature by many design theorists and researchers [Krier, 1988; Wong, 1993; Wong, 1988; Baker, 1993; Flemming, 1990; Clark, 1986; Oxman et al., 1987]. These design composition principles are represented in both 2D and 3D forms. In architecture, these principles are attached to other functional values and are affected by construction methods and materials.

Design composition can be described at many levels of abstraction. Design principles can provide one level of abstraction; however, the components of the design composition can be described in a lower level. According to Wong, basic design elements and transformations are the main ingredients of design form representation and provide a language for describing higher level form design principles. These are

the building blocks for developing form. Wong proposed four categories of the basic design elements [Wong, 1993; Wong, 1988]:

- Conceptual elements: point, line, plane, volume
- Visual elements: shape, size, color, and texture
- Relational elements: direction, position, space, gravity
- Practical elements: representation, meaning, function

These categories constitute different types of compositional elements in form design representation. Wong demonstrated that, through transformations performed on these elements, a variety of design forms can be generated. Similar basic transformations are presented by other theorists [Ching, 1979]. Gargus lists: rotation, reflection, shifting, shearing, displacement, variation, reversal, scaling, inversion, recursion, nesting, and hierarchy as basic transformations of forms [Gargus, 1994. Higher design principles, such as symmetry, balance, emphasis, rhythm, proportions, and the sense of space or motion, can be created from these basic design elements and transformations. Repetition of shapes along a curved line can imply motion and identical compositions reflected about an axis can provide symmetry; most design composition principles can be represented in this way. The representation of form that reflects such principles are in many cases referred to as 'Parti' [Gargus, 1994; Clark, 1986; Krier, 1988]. Kreir provides an extended analysis for form composition in architecture. The process is divided into four main parts: representation, operations, elements, proportions. Operations such as kinking or bending, shifting, and overlapping are among the identified form transformations. Other compositional principles are based on the use of analogies and metaphors and through the utilization of types in form design.

Design principles remain relatively constant over time, but the types of generated compositions change from one generation to another. The emphasis on specific principles can shift, with technology playing a crucial role in this shift. Recently, computers and graphics have revolutionized these types and how they are utilized. Functions are no longer rigidly confined within the constructed spaces; the mobility of the work space and the flexibility of spaces in accommodating several different functions are among recent paradigms in design [Mitchell, 2002]. This has great effect on how the design principles can be applied in the design process and what they can achieve.

In architecture, computer support for design composition has used many different approaches, including shape grammars formalisms [Flemming, 1987], graphics algorithms [Mitchell, 1994; Kolarevic, 1997], evolutionary computations [Jakimowicz et al., 1997], and fractals [Frazer, 1995; Bruton, 1997]. But architectural design practice is still in need of further research in order to utilize computation in practical ways particularly in design composition.

This thesis proposes a technique for computer support for design composition through reusing previous design form principles encoded in cases. This approach develops a language for representing these principles and utilizing them in new situations. It includes a method for representing design episodes or design rationale encapsulated within these principles. These principles are broken down to simple rules, alternatives, and decisions along with the reasoning behind them. The approach uses a CBR methodology to deliver assistance in design composition.

The proposed transformational rules resemble the shape grammars formalism. However, shape grammars do not explicitly provide a language for representing design principles. Although based on the same corpus of geometry, the methods are different. Shape grammars lack the goal-oriented character of design composition presented in this approach. Recently shape grammars approaches use parallel grammars or others techniques to achieve objectives or goals in the process [Knight, 1999].

This approach in this thesis restricts the process of form composition using cases, but the rules here are not automated as in shape grammars. The transformations are provided as operational tools to the designer, who can choose when and where to apply them. Also, grammars usually are developed from a specific set of designs and require great effort to build and test. This approach also provides the capability for the designer to build and extract his own rules developed from cases, which is preferred where innovation is always favored as in the architectural design, as Knight points out:

> …in architecture and the arts, fields with a heavy emphasis on originality and novelty, it seems unlikely that a designer would implement any grammar but his or her own
>
> [Knight, 1999, pp. 4-5]

2.2 Analogical reasoning in architectural design

Architectural design is a complex task that requires high level cognitive resources. Designers often use a set of heuristics and strategies to accomplish this process. Among these strategies is the use of analogy. These strategies differ from design task to another. They are used to overcome difficulties in the design process, such as reaching creative ideas, resisting design fixation, and avoid earlier design failures. Designers rarely start from scratch when faced with new design problems, they usually refer to their experience and build analogies from there. In practice, they are often faced with problems similar or analogous to ones they have encountered and solved in the past. The analogy can be to solutions, the way they derived those solutions, or at least some aspects of those solutions. Past solutions are held in the long term memory and accumulated through years of practice. The process of using analogies in problem solving is called analogical reasoning. Analogical reasoning appears to play a key role in creative design [Bhatta et al., 1994; Schmitt, 1995; Gasakin & Goldschmidt, 1999; Qian and Gero, 1992; Chiu et al., 1997].

In architectural design, designers refer to past experience in many ways; as technical solutions to design problems, as heuristics and techniques to solve the design problems, or as design solutions themselves that provide exemplars of similar classes of design problems. How designers refer to their past experience is not clearly articulated in design research and, in many cases, is considered part of the design process itself. Much research has attempted to uncover the delicacy of this analogical use of flashes of memory and described these exemplars as design precedents, case studies, and others [Bhatta et al., 1994; Goldschmidt, 1995].

Analogy takes place in the designer's mind. To describe the mechanism of how this information is stored and retrieved is related to proposed models of human memory – called memory models - in the field of cognitive science. Among the famous models is Tulvering's model for human memory [Konar, 2000]. This model divides the memory into three parts: sensory memory for receiving information,

semantic memory for conceptual inferences, and procedural memory to hold actions and procedures resulting from the conditions from semantic and sensory memory. In architecture, the sensory memory is responsible for recording information pertaining to how to experience architecture such as seeing, feeling, and others. Semantic memory may hold previous ideas, aspects, characteristics of design solutions and their connections, whereas the procedural memory may hold actions related to the derivation of solutions and actions specifically related to the techniques used in the design process.

Research in cognitive science has found analogy to have a large role in human reasoning, especially for challenging tasks such as planning or design. Analogy also permeates the different techniques for using past experience, such as avoiding previous similar failures, finding new concepts analogous to particular objects, or reusing applicable design precedent. Analogies can refer to natural or human made objects, examples are well known in the design precedents such as by Le Corbusier's habitat that resembles a ship, Utzin's Sydney Opera House that is analogous to a sail or a ship, and others [Goldschmidt, 1995; Schmitt, 1995]. The use of analogy can be conscious or unconscious; designers may unintentionally draw these analogies in their minds.

In architectural design, many ideas and concepts come from analogies from other domains. This phenomenon can be investigated as a possible type of design solution sources. Research in analogy and metaphor has focused on the nature of the mapping process between the source and the target. "Structural mapping" theory has been developed by Gentner [Gentner 1983, 1988], which emphasizes the use of deep characteristics rather than face attributes in the mapping process. Structural mapping engine (SME) [Falkenhainer, Forbus & Gentner 1989] and networks of abstractions and analogies have also been developed [Hampton 1998; Bonnardel et al., 1998].

2.3 Case based reasoning in architectural design

Computer models of analogical reasoning have been proposed to mimic the human thinking process. Analogical reasoning engines have been implemented in many research projects in CS community with good results in specific settings. More details about analogical reasoning can be found in literatures [Gentner, 1983; Kedar-Cabelli, 1988; Turner, 1988]. Traditional AI methods fall short in problem solving in unstructured domains such as design, planning, and diagnosis. A more feasible computational approach is to avoid representing deep domain knowledge and to use the analogous knowledge instead, using cases that represent specific situations and related knowledge. CBR is one of the applications of this approach; however, CBR limits the analogy to a single domain, which makes it a more feasible machine reasoning technique. CBR is a special case of analogical reasoning. Recently, CBR has become a mature strategy in CS community and it has been applied in many different fields, including classification, design, planning, and diagnosis [Juris, 1993; Watson & Perera, 1997; Pu, 1993; Smyth &Keane, 1996]. Since CBR is the main methodology to be used in this research, it will be discussed in more detail in this section.

Much of human reasoning is case-based rather than rule based. When people solve problems, they frequently remember previous problems they have faced. This remembering happens in many situations and reflects our mind's constant search for old information to help in processing new information. We are constantly accumulating new cases and comparing them to the existing cases in an effort to understand

the next case that appears. Often, we make rules or theories from cases and remember the rules instead of cases, but we still need to have access to a wealth of cases from which to understand and generalize.

CBR emphasizes the role of situated experiences. CBR makes analogies between the current problem and previously-encountered situations. CBR can also be used as interpretive approach such as in classification applications. As in analogical reasoning, case based reasoning involves search, match and transfer. It usually starts with describing a problem, then searching for similar past problems. Once a matching problem is found, its solution can be transferred and reused in the current problem. In CBR terminology, these processes are referred to as: index, retrieve, adapt, reuse, revise, and retain, which is the CBR cycle [Aamodt and Plaza, 1994].

The CBR cycle can be described with the following four steps:

1. **Indexing:** Indexing is the selection of features that tend to predict solutions and outcomes of cases. Indexes can be used to construct a case memory and to access cases efficiently. Vocabulary, descriptions, and classifications are used to index cases in the case base.
2. **Retrieval:** CBR Retrieval involves inexact matching of a query focusing on the most useful case to the problem at hand. A similarity measure has to be developed to allow retrieving the most beneficial case. The retrieved case provides a solution to the new problem.
3. **Reuse – Revise** (adaptation): Reuse and revision involve adapting the retrieved case to fit the problem at hand. In general, this process is referred to as adaptation.
4. **Retain:** When a new case is developed after modifying or adapting an old case, the modified case has to be retained and stored in the case base for later use. This new case should be indexed in the same way as the original cases.

Early applications of CBR were developed around the concept of providing memory as a repertoire of cases or situations and organizing them for effective retrieval. Memory models are the subject of much research in cognitive science. Early models such as memory organization packets (MOPS) were developed by Schank [Schank, 1982]. Recent knowledge representations and encoding techniques for memory models include frames or classes in the object oriented paradigm.

Initially, CBR was thought of merely as a memory model to help in remembering previous situations. CBR now is a methodology for problem solving that can be used in wide range of applications such as design, planning, configuration, and diagnosis. CBR can be used as a knowledge acquisition and representation techniques which is extremely useful in less-structured domains. Knowledge is represented through coherent chunks called cases without representing relevant or deep domain knowledge. CBR provides several knowledge containers that can help in the modeling of application domain. These knowledge containers can be found in the cases and their contents, the vocabulary used to describe and index cases, the similarity measure used when matching cases, and in the solution adaptations or transformations [Lenz et al., 1998]. The use of CBR in design is generally referred to as case based design (CBD). However, the applications of CBR in design can at different levels of assistance. It can vary between providing only explanations, indexing and retrieving of design solutions, or being used as a complete design problem solver. The CBR cycle can be completely or partially developed within an

application. For example, many applications stop at the retrieval of solutions, leaving adaptation for the user, other applications may place less emphasis on retrieval and provide complete adaptation methods [Watson and Perera, 1997]

2.3.1 Cases and case representation

The purpose of cases is to represent experiential knowledge, whether this knowledge is about false or true, wrong or right, useful or not etc. A case is a recording of an episode where a problem situation was totally or partially solved. A case should represent such episodic coupling of problems and solutions. The process can be described as a function between domain P (problems) and domain S (solutions) where cases are the pairs (problem, solution). The effect or feedback of this solution can also be detected and added to the case. Cases are represented as simple vectors of these pairs, sometimes called attribute-value pairs or structured representations as classes of objects with attributes and behavior. In many situations, evaluation of the solution is provided with the case. A domain of evaluations (E) can be added and the case will be the triplet (P, S, E) or (P $\rightarrow$ S, E). In the case base, where these cases are stored, indexed, and retrieved, data structures are used to hold and access these cases. Data structures can be as simple as a flat list of pair-represented cases or equipped with additional structures, such as decision trees or other object oriented methods.

2.3.2 Similarity measures and case retrieval

Retrieval is a basic operation in case bases. A query to a CBR system presents a problem and retrieves a solution using inexact matches with the problems from the cases in the case base. The inexact match is based on similarity assessment between the new problem and cases in the case base using similarity measures. As described in [Borner, 1998], there are two different approaches to similarity assessment in CBR; the similarity (or computational) approach and the representational approach [Kolodner, 1993].

In the similarity approach, cases are stored in an unstructured way and retrieved based on measuring the similarity. The usefulness of a case is estimated based on the presence or absence of certain features. Similarity is assessed through numeric computation and results in a single number e.g. weighted sum, which is intended to reflect all aspects of the similarity. Strategies are proposed to reduce the complexity of structural comparisons such as multi stage filtering or mapping. In the representational approach, the cases are pre-structured and retrieved by traversing the index structure, e.g. memory organization packets (MOPS) [Schank, 1982]. The similarity is assessed based on the location of the case in the indexing structure; neighbors are assumed to be similar. Other indices in the memory help in retrieving the useful cases. Some techniques combine these two fundamental approaches, such as *kd-trees* and Case Retrieval Nets (CRNs) [Lenz et al., 1998].

2.3.3 The adaptation process

In adaptation, the retrieved cases are subject to revision and adjustment to fit the new situation. Adaptation is considered one of the most difficult tasks in developing CBR systems, because it requires modeling of domain knowledge. This violates the premise that CBR minimizes the need for deep domain

knowledge modeling. Many CBR applications either skip this process or develop strategies to reduce its impact on the system [Leake, 1996]. There are three general kinds of adaptation [Cunningham and Slattery, 1994; Kolodner, 1993]:

- Parametric adaptation: such as substitution, instantiation, parametric adjustments
- Structural adaptation: similar to the transformational analogy approach [Carbonell, 1985]. It adaptation operators to adjust the retrieved solutions (such as rules or grammars) or model-guided repair and others
- Generative adaptation: such as derivational analogy [Carbonell, 1985]. It reuses and adapts problem solving episodes through replaying (derivational replay)

2.4 Architectural typology and CBR

Type in architectural has a broad meaning and carries significant knowledge for the design product, the process, and the relations among elements. Typology is the enumeration of elements with their adjacency and other relations. The term "Typology" has different meaning depending on the context in which it is used. It may serve as a definition for classifications, design knowledge containers, or models of designs or design solutions. Different kinds of typologies have been utilized in the field of architecture such as functional typologies, formal typologies, and others. Typologies in design allow the transfer of known solutions characteristics to new related architectural problems. In this way, typology serves as a basis for analogical reasoning [Leupen et al., 1997].

Typologies can be classified as: building types (models), organizational typologies, and elemental typologies. *Building typologies* represent the specific characteristics of a particular class of buildings such as schools, hospitals, or churches. It includes aspects related to that particular type of building such as needs and uses. *Organizational typologies* provide a general framework for solving problems related to the spatial distribution or the order of functional elements. It involves topologies, spatial arrangements of functional elements, and rules or patterns for formal composition. Layouts with courtyards, U shapes, or L shapes are examples of such organizational typologies. *Elemental typologies* refer to prototypes for solving general classes of design problems. They are generally related to the design of elements of buildings, for example, the entry to a building, the type of vertical movement, or the level of privacy required. This classification of typologies is not inclusive; one building or part of a building can serve as a prototype under all the three categories [Rowe, 1987].

Typologies play a crucial rule in architectural design [Leupen et al., 1997; Gargus, 1994; Norberg-Schulz, 2000; Gero, 1990]. Many designers restructure their thoughts and explore different design solutions use typology as a strategy. The concept of typology can be used through different levels of detail and abstractions. For example, a school building is a subtype of educational buildings and a classroom is a subtype of space and a six-foot, double-hung door is a subtype of opening. Topology is high level geometry [Stevens, 1993]. A topology is a specific abstraction and can be incorporated within types and typology. For example, the topology of double-loaded corridor in hotel design is an abstraction to the form of the hotel and can be used as another classification to which types and subtypes can be

linked. So, the concepts of topology can provide an effective classification for design cases and their components.

Typologies are often used in design computation. Building models are used in model-based reasoning. Elemental typologies provide types of design problems that can be expressed as stories or situations. Organizational typologies are central to layout problem representations (the space allocation problem). However, typologies can also be used to support design composition as knowledge containers, as alternatives, and as sources of composition to guide the design process.

In CBR, cases can be generalized patterns or configurations, which are similar to the patterns found in organizational typologies [Leusen, 1995]. Classification of cases is crucial to CBR since it is part of the problem solving process itself. Organizational typologies are ideally suited for CBR for design composition through the use of classifications related to the geometrical configurations of cases. Some CBR systems have used narratives about building elements and situations of design problems such as "design stories" in Archie II system [Domeshek & Kolodner, 1992]. Such systems utilize elemental typologies in their CBR.

Typologies implicitly contain both iconic and canonic qualities of literal analogies. This connects the CBR methodology to the architectural typologies. The difference is that these analogies are to previous solutions that have been tried and found useful rather than new untested analogies. Both historical examples and current practice are good sources of cases. Organizational principles and typologies stemming from current practice, such as circulation patterns, can provide roadmaps for new solutions.

Finally, the effectiveness of any heuristic in design depends on its appropriateness to the problem at hand. Consequently, both the timing and the contents of the typology are crucial to the process. In CBR, this can be translated as effective and efficient retrieval mechanism. The retrieval is highly dependent on the vocabulary and classifications, which highlights the importance of the proper choice of typologies for the CBR system.

2.5 CBR and building design

Building design has special characteristics that differ from other design fields. Within building design, there is a wide range of activities with many different design problems and solutions. In the early phases, the process focuses on developing a preliminary list of functional requirements and a conceptualization of the building form. Throughout the process, many architectural design problems must be solved, such as site, form, environment, materials, and structure. Given this diversity in the architectural field, computer assistance should be specific to the kind of problems being addressed and the expected solutions. CBR has been used to solve a variety of problems from interpreting or explaining a solution to automated solution generation. To successfully develop a CBR system, determining the specific problem and the nature of the expected solutions for that problem is crucial.

CBR has been used in many architectural design systems. REALTOR is a simple system for the evaluation of real-estate properties, based on prior experience in appraisal and sales [Cunningham and Slattery, 1994].

Archie II and DOORS are computer assistance systems for conceptual design. They provide indexing and retrieval capabilities of many architectural resources. Archie II allows retrieval of recorded design stories describing design mistakes and possible corrections. It uses a bubble diagram representation of design cases [Domeshek & Kolodner, 1992; Domeshek et al., 1994]. DOORS allows for recording of browsing sessions for later use [Sklar, 1995].

HouseCAD is an early design assistance system for housing. HouseCAD assists design brief development and conceptual design. In HouseCAD, a set of cases can be retrieved that fulfill the aspects of the desired brief. HouseCAD uses a conceptual representation of cases (bubble diagram) to index and retrieve cases. It uses the relative room positioning, coordinates, and orientation to represent the basic spaces within the house. HouseCAD relies on the dynamic cycle of selection and adaptation of the house diagrams, represented floor plans. HouseCAD is helpful in the early phases of the conceptual design, primarily for the functional layout design problem [Raduma, 1999; Raduma, 2000].

Oxman proposed several systems for precedent based design and memory management such as MEMORIBILLA and PRECEDENT [Oxman, 1994]. These systems use the approach of contents and story-based representation by decomposing designs into tractable pieces of information. These systems provide a CBR methodology for representing concepts and stories as cases, but do not provide representations of the design composition aspects.

CADRE (ACABUS) is a design assistance system for conceptual design. It goes beyond indexing and retrieval in allowing elaborate adaptations for the retrieved cases. The cases are represented as 3D models with views of structural, architectural and mechanical characteristics. Topological adaptation and case combination are supported as adaptation strategies with less emphasis on the indexing and retrieval [Schmitt, 1995; Schmitt, 1994].

Kuhn and Herzog apply design patterns and a language game abstraction (LGA) to the design process [Kuhn & Herzog, 1994]. Cases provide links between the configuration phrases and instances that satisfy them. For example, a link between ‘concentric space’ and ‘dome’ can serve in the retrieval of several useful cases. The approach relies on textual abstractions of cases and uses a language game metaphor in representing and retrieving design cases. The LGA approach can provide a rich environment for exploring design solutions including form compositions, but it focuses on retrieval of cases. Designers must decompose the case, as in the traditional design process, in order to make use of the embedded design concepts. What is relevant to design composition is the focus of LGA on configuration aspects and the isolation of them from other contexts. This can provide rich selections and descriptions of design composition components, concepts, and methods.

SEED is a CAD system for wide range of design problems including, requirements development, layout design, and 3D enclosure modeling. SEED provide a clear representation of design problems and solutions and utilizes CBR technology to manage the resulting solutions for later use and adaptation. SEED adaptation is limited to the editing capabilities used in the generation of solutions [Flemming, 1994a; Flemming et al., 1997]. FABEL is a design assistance system for technical installations. It utilizes previous installations to create ones in the new buildings. FABEL uses a modular case representation that allows adaptation of prior solutions, called Armilla 5 [Schaaf and Voss, 1995]. CADSYN is a structural

design assistance system that helps in designing structural system for high-rise buildings based on previous designs [Maher, 1993]. Rivard investigated the reuse and adaptation of building technologies, within cases, in conceptual structural design [Rivard & Fenves, 2000]. Design composition is addressed in the SEED configuration module, in which a well composed outline is converted to a 3D mass. SEED uses first principles in the generation of the layout design and relies on the CBR technology to keep track of the generated solutions. External solutions can be entered using the SEED representation, but this is not the intended way to use the system. SEED has no representation for the concepts used in the composition itself, such as symmetry, gradation, rotation etc.

2.6 Summary and implications for this research

New approaches are needed before CBR can be utilized as a process model for design composition. Explicit representations (called knowledge containers in CBR) of the vocabulary of design composition and the process itself are required. In addition, formalization of many aspects of design composition for CBR is lacking, i.e. vocabulary for indexing, operations, transformations, and adaptations.

Although CBR is a mature technology in other fields, in architectural design composition it is still in need of development. Many aspects of design composition are suitable for case representation and problem solving. Also, the capabilities of CBR can effectively aid in finding innovative solutions or design compositions.

From the review of CBR approaches in design composition, these general observations can be made:

- The definition of design composition as a problem is imprecise; many aspects of the early processes are included in design composition, such as brief development, layout design, and others.
- Aspects of design composition in cases are not made explicit beyond the linking to architectural descriptions and classifications
- In many cases, CBR is used as a utility within a larger system, not as the main reasoning engine focusing on a specific problem and solution.
- In design composition, many problems can be identified as suitable for CBR, but difficulties arise in finding a representation for these problems and their solutions.

In this research, we consider precedents and their use from a computational perspective. We start with defining what these precedents are and how they are used in the design process. We then investigate how to represent these precedents and how to use them in this representation. We use the term ‘design precedent’ in its architectural sense, that is, as a prior design that has some interesting architectural characteristics for designers to refer to. These characteristics could be new building concepts, organizations, physical characteristics or other intangible characteristics. We are concerned with building a repertoire of these precedents in an efficient way to allow finding and making use, of them. To accomplish this, we need to address some requirements: To identify the precedent characteristics and to make sure that the collection covers a variety of designs so that it is beneficial for architectural designers. These precedents are documented in several types of formats. We need to accommodate these formats, such as images and CAD drawings and to preserve those characteristics which are dependent on the

format. We need to understand why these precedents are interesting to designers. We need to elucidate and isolate these interesting characteristics to make them available to designers, so that they can retrieve them precisely and efficiently. This must be done carefully to keep them in context and not to loose their inherited characteristics. We need to know how designers will use cases in their design process, in order to develop representations that enhance their use. Finally, from the computational perspective, we need to make sure that the representation is not so complex that precedents will need too much time and effort to be used. It is hard to develop a representation that satisfies all these requirements.

The following aspects form the design precedent:

- Functional aspects
- Elements, spaces, circulation
- Organization of functions & hierarchy
- Form aspects
- Geometry and shapes
- Organization
- Ornaments, elements
- Massing
- Texture, materials
- Technological aspects
- Construction methods, materials
- Environmental aspects

3. Research problem / Motivation / Objectives

3.1 Research Problem

The research problem can be divided into three components: the lack of computational support for the early phases of the design process particularly design composition, the need for better computational methods that address core problems of design composition, and the problem of adaptation of prior cases in a manner that is feasible and useful for design composition.

3.1.1 Support early phases of the design process

The importance of the early phases of the design process cannot be underestimated because the quality of the design depends on the decisions made at this phase. For example, in architectural design, continued development of an infeasible concept will not improve the quality of the result. On the other hand, a sound design concept would likely be propagated through the design development and encourage better qualities.

Integration with later design phases is needed. In most cases, when concepts are developed using pencil and paper, it is hard to transfer the concept to electronic media without loosing the flow of thought or the creative impulses. On the other hand, easy and fast simulations and analyses are available in computational media, which are not available using pencil and paper. Integration can happen only if the gap is closed between these design phases. Concepts and their development must be computationally supported. The design development phases have to be more receptive to the soft computation needed for

the early phases. In addition, using computation in the early process can enhance the possibility to reach creative solutions through the examples and precedents available in this media. Computation and mathematical formalisms can also provide other venues for creative leaps.

3.1.2 The need for better suited computational methods

Until recently, the majority of designers had their experience in pencil and paper design, particularly at the early phases. It is hard to change their practice unless CAD comes provides accommodating methods and interfaces. There is a need for flexible CAD support that allows designers to use computers as partners at this early phase. Design and usability of CAD software need to be rethought if designers are to actually use CAD in their early design sketching. The goal is not to make computers mimic human designers; on the contrary, it is to allow computers to play their role integrated with the design process and to work human designers. It is imperative to develop representations that can be understood by both designers and computers in the early phases of the design. Such representations will allow integration between designers and computers and enhance the use of computational methods.

Within the computational community, knowledge acquisition and representation are considered the bottleneck for many applications. Supporting design requires modeling complete domain knowledge, which is not feasible. AI methods and techniques can bridge this gap but until recently, such methods were not applied to complex fields such as design and were targeting less complicated tasks. There is a need for extending AI methods and techniques to support design tasks [Flemming, 1994]. This includes methods in knowledge representation, reasoning, and collaboration with users.

3.1.3 Adaptation in CBR to fit the application field

In CBR, adaptation is considered one of, if not the most, difficult problems. Mainly because of the need to represent both domain knowledge and problem solving knowledge, which violates the purpose of using CBR in the first place to avoid such representation. On the other hand, the objectives of CBR, in many applications, cannot be completely achieved without adapting the retrieved cases. This kind of conflict has troubled the CBR field for long time. New models that build on CBR and accommodate effective and feasible ways of adaptations are needed. Researchers have suggested many techniques to deal with the adaptation problem, such as re-applying CBR methodology, the use of adaptation cases that hold similar examples, adaptive retrieval, derivational adaptation and others. These techniques need to be integrated into CBR and tailored to the meaning of adaptation in the application domain [Leake, 1996].

As noted, the adaptation process is much related to the specific domain that CBR is applied to. In architectural field, the adaptation process cannot be precisely defined or described. Examples of CBR systems in the architectural field either avoided the adaptation process or dealt with it in a very specific context. Therefore, there is a need to solve this problem through suggesting models for CBR that provide effective adaptations that match the architectural reasoning process.

3.2 Motivation

Analogical reasoning and CBR match to a large degree the architectural problem solving, particularly in the early phases. Cases, precedents, and analogies are widely used in this process.

Analogies to different domains are drawn in many situations. This match is the motivation to explore how this technology can provide computational support for the early conceptual design process.

Many researchers focused on the early design process primarily looking at resolving functional conflicts or building topology. In architectural design, form must be sketched before embarking on a detailed solution or evaluating solution. Form selection or generation is the core of the process. The goal of this research is to provide computer assistance for finding design forms.

3.3 Research Objectives

This research uses CBR as a generative technique for design composition. Through the reuse of cases, new forms will be generated. Form generating and reusing previous forms in design have to be considered in the CBR approach. Instead of reusing the solution itself, the generative approach abstracts and back-tracks the possible path of the solution. The solution path includes the decisions, methods, alternatives, and special processes that were used to shape the design form. At the heart of this approach is the use of derivational analogy concepts [Carbonell, 1985], which focus on detecting similarities in the derivational path rather than the end solution. This derivational path, if reused in delivering new solutions, could be described as a generative path. So, the approach is generative CBR, or G-CBR. This approach requires representation of both the end solution and the derivational path.

The main objectives of this research are:

- To represent the design composition problem as accessible and usable constructs through a visual language for abstracting the design intent or rationale
- To include the main functional characteristics in the representation, since architecture is not only about shapes, but also about function
- To make intensive use of precedents in design, since this is the way designers use their experience
- To resolve the difficulties in utilizing CBR technology in the design field, particularly in adapting solutions, by using a model that relies on derivational analogy to capture design intent and to replay and regenerate new designs.

4. Research approach

This thesis uses CBR and derivational analogy as a problem solving methodology for architectural design, particularly for design composition. A new construct is proposed to represent the previous path that led to a design solution. This construct is called Sol-Traces, for solution traces. Derivational analogy and Sol-Traces are encapsulated within CBR functionality as the main reasoning. This research is restricted 2D design compositions and forms. The approach deals only with the two-dimensional aspects of design compositions and forms. The research hypothesis is that architectural design compositions can be represented using a language of geometric form and Sol-Traces constructs, and that this representation can be modeled computationally.

The research builds on research and analysis of design composition in architecture. The approach uses analysis, constructs, and principles of design composition process outlined in many sources [Krier, 1988; Laseau, 1989; Lawson, 1997; Rowe 1987; Flemming, 1990; Clark, 1986]. However, this kind of

analysis, constructs, or principles did not find its way in a computer assistance system for design composition. Therefore, the main objective of this research is to provide computational support for this process using Sol-Traces and the generative CBR methodology.

4.1 Sol-Traces representation

This thesis proposes the use of Sol-Traces which is a virtual construct that represents the solution path or the route that can lead to a solution for a particular problem. This path can be used to solve similar problem in the context of the original problem. The concept of traces originated in the derivational analogy model of problem solving where the analogy is to the process of deriving the solutions. Carbonell noticed that in the analogical problem solving approach, similar problems may not have similar solutions but, the solution paths were similar and this similarity in the derivational path could be used in problem solving where traditional or direct analogy could not provide assistance. The derivational path includes all the aspects of the reasoning process used in deriving the solution, including methods, alternatives, decisions, and justification. In theory, this path, if applied in a new problem, will yield valid solutions [Carbonell, 1985]. Figure 4.1 shows the basic difference between the traditional CBR model and the model using derivational analogy. The main difference is that the adapted solution is reached through the original solution in the traditional model, whereas in the derivational analogy model the new solution is reached by reusing the derivational process without information about the old solution. The new solution in the derivational analogy model differs from the old solution, but share the derivational process.

Sol-Traces representation allows for new suggestions, flexible restructuring, and alternatives using the solution path. If such path was extracted and followed in new circumstances, it would generate a valid solution or solutions to the new problem. Extracting and representing such a path requires identifying the path along with all alternatives, decisions, methods, and the reasoning that resulted in this path. We need to represent the knowledge related to the design problem / solution characteristics (domain knowledge) that allows inactivating the trace in the new circumstances. We need to build a system that can read the path from one problem, apply it within the characteristics of the other problem, and reach correct decisions and alternatives using the encoded reasoning in a flexible way. However, this is harder than representing the design problem and the knowledge required to solve it, which eliminates any benefits, gained from using the case based reasoning approach. The research approach has been developed to alleviate these difficulties without loosing the benefits of the derivational analogy as described by Carbonell [Carbonell, 1985].

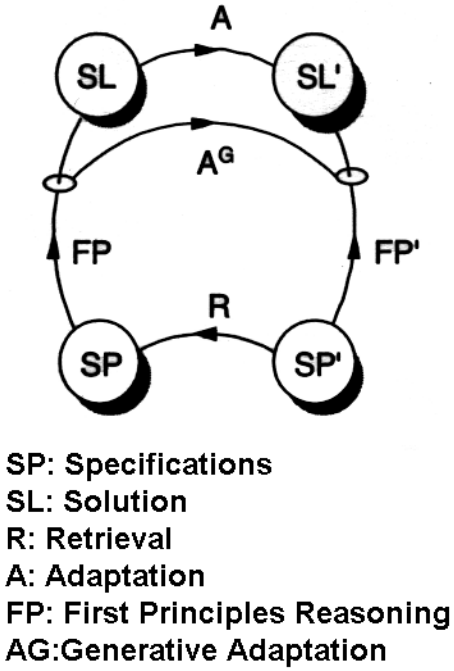


Figure 4.1. Comparison between the traditional CBR and the generative (derivational) CBR [from Cunningham, 1997]

The approach differs from Carbonell description in easing up the requirements for structuring these traces and the reasoning encoded within them. There is a semi-representation of that reasoning or light-weight traces. It is a shared approach between the modeled reasoning and the designer's actual reasoning and interaction. The derivational path – solution trace – is to be represented as a construct that carries, as much as possible, of the reasoning process and can be reused (replayed) in order to be converted to as solution for the new problem. This reuse or replay will be achieved through the possible adaptation/generation mechanisms within the trace, under the control of the designer. This will be carried out on a way that allows different unpredictable solutions and generation to happen and surprise the designer with possible solutions. Similarities between the derivational processes will allow them to be modeled outside the derivational trace itself instead of repeating these processes in each trace. This will make the traces less complex whenever possible.

The suggested representation of derivational path or "solution traces" can be divided into two categories: static representation and automated (dynamic) representation:

A. **Static representation:** solution traces are represented as static constructs, they can be adjusted, modified, or adapted, but they remain static in the sense that they will not be reconstructed or regenerated. The reasoning is stored in the traces as a source of possible alternatives, parameters that can be changed, or constraints and characteristics of the elements within the trace. The designer's input or reasoning is required to complete the process, such as choices or parameters

and the reason behind using them. Sol- Traces will be adapted through several processes and new solutions for the new problem will be reached through operations such as:

- changing parameters
- adding or deleting elements
- changing locations or topology
- changing constraints or relations

B. **Automated representation:** Sol-Traces are represented as constructs that can be reconstructed (regenerated). This category provides deeper reasoning support, which is used in guiding the replay of the trace within the new problem circumstances. The goal is to have interesting results that are not always predictable by the designer. The regeneration is used to reach a suitable solution for the current problem and usually executed to propagate or accommodate changes such as:

- using new parameters, shapes, or location
- accommodating changes in constraints
- replaying with a change in the order of processes, such as sequential, nested, or others combinations

In the following sections, Sol-Traces are presented with their sources in the architectural domain.

4.2 Sol-Traces in architecture

Design in architecture involves a wide range of problems and solutions. In order to successfully apply the concept of Sol-traces, a specific class or type of problem and solution must be addressed. In this research, the problem is limited to form generation or design composition. The problem specifications will be the functional characteristics (or needs) and the solution is the assigned form. This problem – solution definition can be applied to different levels within the architectural design domain, such as the design of a building, a part of building, or a group of buildings. In architectural design, designers usually use several different strategies to generate forms. They master some and continuously learn others throughout the practice. These strategies consist of a set of tactics, plans, guidelines, or heuristics related to the generation of forms or finding these forms.

Among the important ones are strategies related to the reuse or learning from previous solutions and the methods or techniques involved in developing them. These embedded methods and techniques have to be identified, extracted, and then adapted to solve new problems and usually results in a series of building forms that share some intrinsic characteristics; either generated by the same designer or by other designers who adopted a similar strategy. In this research, the strategies, the methods, and techniques, as well as the reasoning they carry are used as basis for building Sol-Traces, combining the representation of strategies for reuse and making use of other strategies found in designs as traces or paths of the solution. In general, these strategies can be divided as follows:

A. Strategies related to analogies and metaphors: such as direct or indirect analogies to different kinds of objects or relations

B. Strategies related to shape, geometry and their relationships: shapes such as circle, square, triangle and relationships such as repetition, similarity, gradation, hierarchy etc.

C. Strategies related to the functional characteristics: such as differentiating between functional space and circulation space, type of circulation pattern and others

D. Strategies related to type of architectural organization: such as centralized, linear, radial, clustered, and grid

E. Strategies related to the use of architectural elements, such as roofs, mass, doors, openings etc.

This research proposes to develop a specific language to capture the essence of these strategies and to be used in representing cases in a way that allows these strategies to be accessible for later reuse and adaptations. These strategies carry the reasoning as rules. They are applied in cases which provide realizations of these principles or rules. The approach here is to use this concrete application (cases) and reuse it in similar situation to provide solutions to design problems without requiring thorough investigation of these rules and details of circumstances for these applications.

The proposed language is based on theory work of Wong [Wong, 1993] in his series of books about principles of form and design. Wong presents a visual language to interpret design. Wong's approach is on the formal side of the visual aspects of design; the language and interpretations are based on systematic thinking with very little intuition or emotion. The proposed language is a subset of Wong's visual language elements and operations that are related to form composition and applicable in the architectural design domain. This language provides an interpretation of the design solutions using basic shapes, geometry, and operations, such as a square as a shape and gradation of squares in the plan as a design strategy. The interpretation is an abstraction or reduction of the conceptual idea (or strategy) behind the design solution represented in simple geometrical shapes and transformations. This abstraction provides the basis for the suggested approach or model for design composition. These components of the language are refined, combined, and further arranged to provide techniques for design composition. The representation is used within the CBR approach to index, retrieve, and adapt these components or their resulting arrangements.

4.3 Case representation and Sol-Traces

The proposed representation can be classified in many different ways. Based on the type of characteristics that can be extracted from the case, there are two main categories of traces; functional representation, which deals with the functional characteristics, and the form representation, which deals with form characteristics. Sol-Traces are classification based on the usability or the adaptation model of the traces. Traces can be classified into three main types; the simple category, which is related to the simple representation and adaptation, the transformational category, which mostly follow the transformational adaptation model, and the derivational – or generative category, which deals with the derivational analogy model of adaptation. Other classification can also be suggested, such as the dynamic and static types of traces or others to distinguish between the various traces characteristics in this approach. It is worth noting that these categories are not exclusive, one trace can carry multiple characteristics from different categories.

4.3.1 Functional attributes representation

A floor plan is a rich representation of many functional aspects of the design as well as many other technical aspects. No and the design concept are:

1. **Circulation and use elements**: Spaces are characterized either as circulation space or use space. This abstraction of the function of space plays a crucial role in the composition of a design solution and provides a design decision and justification. Circulation or use elements can have several patterns such as linear or circular. Figures 4.2 shows representation of the circulation and use attribute

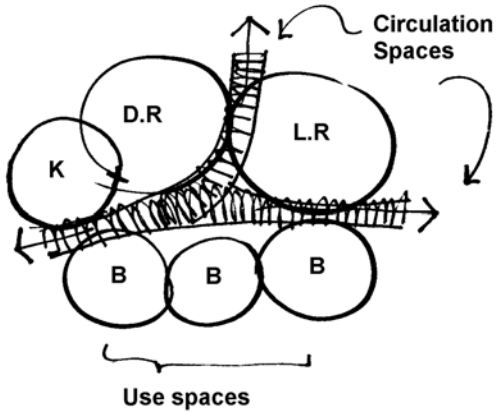


Figure 4.2. Representation of circulation and use attributes

2. **Enclosure types**: The type of enclosure can be represented visually in the floor plan. Krier identified three main types: solid walls, skeletal enclosures, and virtual enclosures Also, a type can be combined [Krier, 1988].
3. **Solid and void mass**: In many cases, the floor plan does not explicitly convey the composition. Distinguishing between solid and void masses can elucidate the design composition. Therefore, the use of solid and void mass must be included in the visual representation. Figure 4.3 shows an example of solid and void composition.

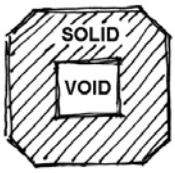


Figure 4.3. Solid and void representation

4. **Areas and adjacency**: The areas and adjacencies of spaces in the design are abstracts of the simple geometry that reflects the design elements. Their location preserves their adjacency in the floor plan. Figures 4.4 show examples of areas and adjacency representation.

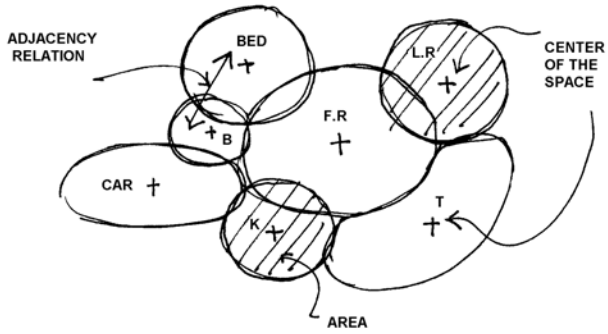
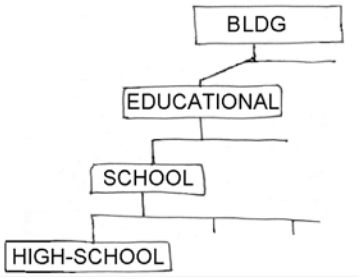


Figure 4.4. Areas and adjacencies representation

5. **Functional hierarchy and typologies**: Two main classifications are proposed: functional hierarchy which represents zonal decomposition of the design regarding the size and grouping of units, and the functional classification of types without considering any form characteristics. The functional typology is based on a building model that can be developed from classifications of building typologies such as educational, health, housing etc. The components of the building model are based on a generic decomposition of building into spaces, zones, rooms, etc. Also, other classifications such as typological aspects of spaces can be used such as circulation space vs. use space. Figures 4.5 shows example of functional hierarchy representation.



Figures 4.5 Examples of functional hierarchy representations.

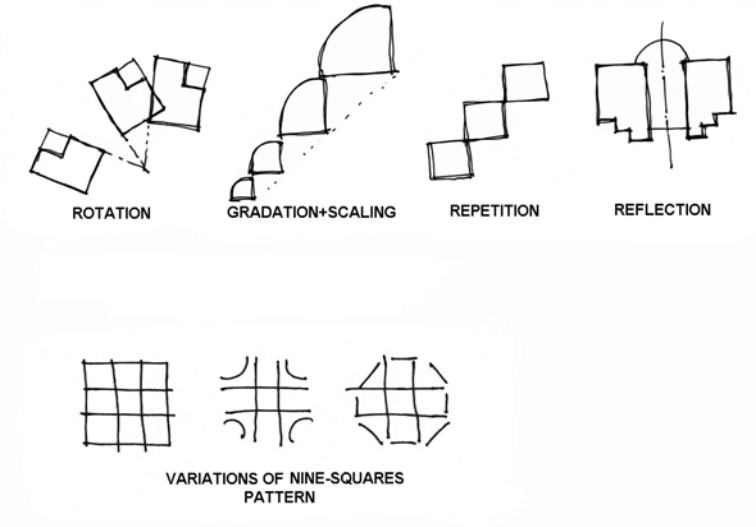
4.3.2 Form attributes representation

In architectural design, form composition aspects are formulated in the early phases of the process and carried over and refined throughout the entire process. Reusing the form composition aspects in a new situation can provide the potentials for new design composition. Form attributes can be visually abstracted and represented from a design, e.g. a floor plan. Three categories of form aspects are proposed, the form types, shape and geometry, and form-function associations.

1. **Form types**: This category captures the architectural form organizational scheme such as linear, radial, or circular. These typologies are different from building function (type), but there is overlapping in many cases. Also, these typologies can be applied to different levels of the hierarchy of a building, i.e. the whole building form, partial form, or the form for a particular element. The form typologies are very open issues; various typologies can be developed. The approach will use multiple classifications of design forms regarding their geometries, organizational, topologies, and other aspects to develop a forms typology model.

2. **Shape and geometry**: This category provides a rich repertoire of form elements and transformations. The basic components in this category are: simple attributes of shapes and geometry, such as a square or a group of virtual compositional lines, the transformations such as repetition, gradation, or kinking, overlapping etc., and the architectural constructs such as symmetry lines, axis, patterns, modules, etc. Figure 4.6 shows examples of this category

Figure 4.6 Examples of shape and geometry representation



3. **Form - function associations** This category of form attributes captures the relationship between the building function and its form (or part of the building). It provides that this type of form is mostly used to accommodate this type of function(s). This category can serve as a recoding mechanism of the designer's experience in choosing and assigning forms to functions. Figure 4.7 shows example of form-function association.

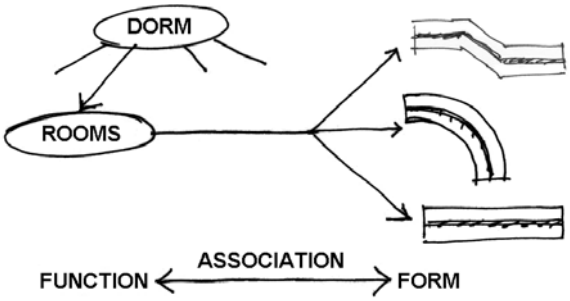


Figure 4.7. Example of form-function associations

4.3.3 Sol-Traces based on adaptation model

Traces can be used in three ways: simple model of adaptation, which deals with simple adjustments and editing or null adaptations, the transformational adaptation model, and the derivational adaptation model. The focus in this research is on the derivational analogy model. All traces are based on this concept; however, the type of adaptation provided may differ.

1. **Simple adaptations**: In this category, the Sol-Traces require minimal adjustment to fit the current problem. This approach is similar to interpretive CBR systems used in applications such as classification and diagnosis. In many cases, there is no need for adaptation since the retrieved Sol-Traces provides a ready answer or a solution to the design problem.
2. **Transformational adaptations**: Transformational adaptation uses the transformational model of analogy. These adaptations can be: parametric adjustments form elements, shape transformations adjustment etc
3. **Derivational adaptations**: The derivational analogy model uses replay and regeneration of Sol-Traces. Parameters and other aspects of Sol-Traces can be adjusted and the trace then be subjected to 'replay' to accommodate the changes. The result is not always predictable, which is a strategy for exploring the potentials of traces. The replay can be as a sequence of processes that can be interrupted and reassembled. Figure 4.8 shows an example for the replay under new circumstances.

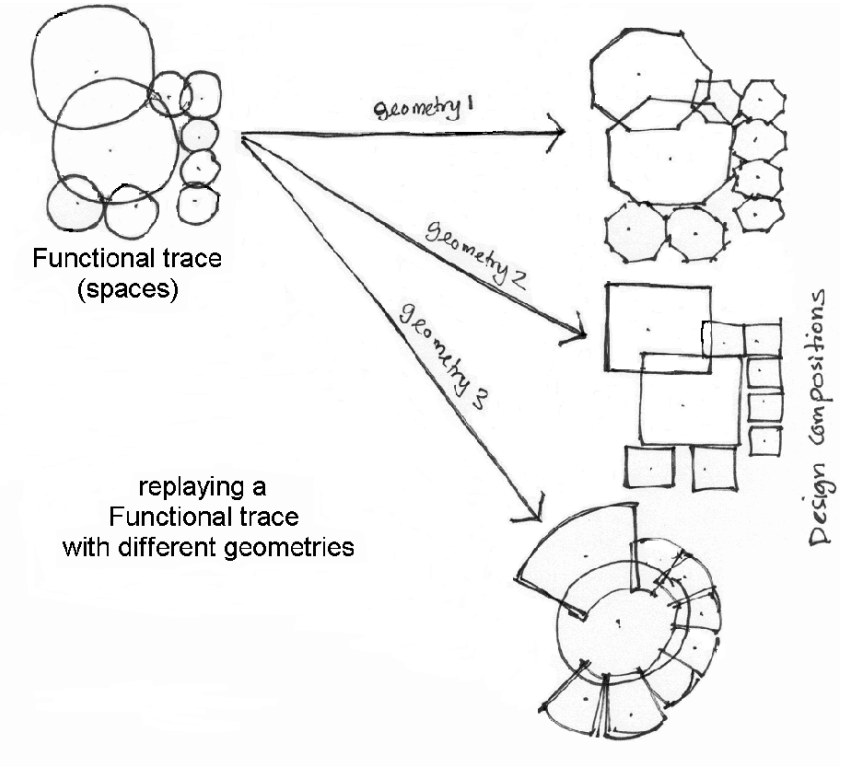


Figure 4.8. Example of the replay process under new circumstances

4.4 CBR using Sol-Traces representation

In the research approach, the Sol-Traces representation in CBR is based on a derivational CBR model, where the similarity is to the solution trace rather than the end result characteristics. The adaptation relies on generative or derivational traces. Starting with a problem and using problem specifications, a matching case can be retrieved based on the problem specifications. The trace of the solution in the retrieved case is used to derive a new solution. Less knowledge is known about the end

solution in the retrieved case, the solution trace is used to guide the process of reaching a valid and desirable new solution. In the generative CBR model, knowledge about the end solution can disregarded and the focus shifts the path of solution generation. For this thesis, solution traces are the main cases in the CBR system, while other information about cases provides support for the CBR functionality. This approach takes advantage of both models of CBR; the robust traditional CBR matching and retrieval mechanisms and the generative characteristics of generative CBR.

Sol-Traces are drawn and the desired objects represented using an Object-Oriented (OO) system development technology. These objects are indexed in the case, as tags for the objects, using the previous attributes (function and form). The traces and their components are indexed either in a flat list using the provided description or as a hierarchy.

For the hierarchical case, Sol-Traces must be added to the tree that represents the general hierarchy (function or form). For example, the retrieval of alternatives or options can be executed through traversing of the tree nodes in the level of the starting option or alternative. Many of these hierarchies or sub-hierarchies will be built through the use of the system.

The reasoning in the proposed CBR approach is divided into two main types: reminding and adaptation. Reminding is the core function of retrieval in CBR; it is based on general characteristics, functional characteristics, and form characteristics. The reminding can take place at any time or at any situation during the use of the system. The initiation of reminding can be controlled by the designer or through automated functions. The reasoning through adaptation can adjust the Sol-Traces in order to fulfill the requirements of the current problem. This adjustment may be editing the Sol-Traces, substituting and transforming, or replaying and regenerating. Sol-Traces must be represented to allow such adaptation. For example, for editing, the representation provides components and parameters to work on. For replay, the representation provides trace components that can take certain behavior based on the new circumstances of the replay process.

4.5 Building the case base

Building a case base Sol-Traces will require a great effort. Relying on a single expert or one group of experts might not be enough to encode a variety of Sol-Traces. A feasible way is to allow this case base to evolve through the practice by having both the system and the users identify, define, encode, and index these strategies wherever they are used. Sol-Traces will be built over time with input from multiple designers. An initial case base with examples of Sol-Traces will be a necessity at the start.

The contents of each case are not limited to the trace itself but can include materials that support the solution and provide quick evaluation. These materials include images, CAD layouts, analysis, architectural illustrations, comments, and others. The flexibility to add such multiple media should be allowed. Sol-traces are indexed using many aspects that can help in retrieving the most relevant, adaptable, and useful ones to the problem at hand. Indexing feature can be:

- Functional characteristics: using the attributes of functional representation
- Form characteristics: using the attributes of form representation

- Sol-traces aspects: related to the type of Sol-Traces or the description of the strategy being used.
- Other data or information within the case
- Other data or information assigned by designers and not included in the previous categories

The system interface should be accommodating for the not well-computer-friendly designers [Borner, 2002; Finger, 1998]. The approach suggests the idea of a workspace, which mimic the designer's place with drawing board, library, and other resources. More than one case can be brought together and the workspace can be saved for later design sessions. Search and retrieval must be interactive to allow for form exploration through multiple steps. More Sol-Traces can be retrieved at any time whenever a designer wants to do so. One retrieved case could lead to further retrieval that the designer did not think of at the start. To assure consistency and to facilitate building the case base, the system should provide templates that can be used to enter the Sol-Traces and make them available for use. Also, adapted sol-traces can be re-entered as new traces. Customized versions of these traces could be developed and recorded by designers, which can be enhanced through out their repeated use.

Because of the variety of indexes used in this approach, calculating similarity measures can be a long process. In this regard, multi-step filtering using the proposed hierarchies of memory structures (form and function) is a feasible approach. Also, the use of OO technology to build the system can facilitate the matching process by providing a filtering scheme. Also, the desired exploratory nature conceptual design puts less emphasis on finding precise matches. In fact, precise matching of solutions is less desirable since the trace will be subject to replay with different circumstances and unexpected results, which can also provide interesting venues to explore.

4.6 Design composition using Sol-Traces

This section discusses the utilization of Sol-Traces in a design assistance system for form composition. The discussion starts with the abstraction process, then the building of the case based system, the reuse and replay of Sol-Traces, and finally more focus on the expected adaptation methods in Sol-Traces.

4.6.1 Abstraction and representation of cases and Sol-Traces:

Sol-Traces abstractions can be applied to the initial set of cases; templates can be developed and used to generate Sol-Traces out of cases. The abstraction/representation process contains three phases:

Phase 1: Sol-Traces are built using instances of the abstraction language

Sol-Traces will be built from abstraction of the design using the proposed language elements and operations. This process can be done by the designer step by step with full control or it can be automated through using templates to capture emergent shapes/forms according to specific settings. These templates can be specific and the combination of them can produce rich representations. In this research, the process of building Sol-Traces will be limited to manual or hard-coded set of cases, along with a set of suggested templates.

Phase 2: Indexing the Sol-Traces: as part of the representation, Sol-Traces are indexed using several components, operations, and other aspects. For example, a Sol-Traces that has a square-shaped gradation aspect, could be indexed with this feature, also several cases with this feature can be linked to this trace. This will allow for efficient retrieval of cases and Sol-Traces.

Phase 3: Advanced representation: advanced representations of forms in Sol-Traces can be developed. Sol-Traces components can be used in an exchangeable format between cases; these traces can be assembled in a sequence form to formulate other larger traces. Also, traces can be set up to receive parametric changes when adapted. Components of Sol-Traces must be synchronized together to achieve a desired output. Another characteristic of this phase is capturing the associations between form and function in the design composition process. These associations can be on different levels e.g. the building as a whole or partial zone, cluster of spaces, or a form that a single space might be associated with. These associations can be indexed and stored in the system memory and retrieved whenever a similar situation is encountered. The system will act as a designer's memory in recalling these associations and reusing them. These associations can be used to guide design explorations and to find the relevant or most beneficial cases. These associations can be suggested by the systems for solving a particular design problem. Future use of these associations can be in the automation of composing a form for a particular building.

4.6.2 The retrieval of cases and Sol-Traces

The retrieval process is as follows:

- Query the case base Sol-Traces using predefined characteristics
- Search while adapting case i.e. matching a sequence of adaptation operations
- Search for form characteristics associated with a certain building or a design problem.

Searching, matching, and retrieval are the basic reasoning, and problem solving process in the CBR methodology. Best matching Sol-Traces will be retrieved for a set of descriptions. These descriptions could be related to the function, the formal characteristics, or other attributes of cases and Sol-Traces. Users can also retrieve more cases by adjusting the retrieved Sol-Traces and searching for similar cases. Adaptation-guided retrieval happens when a case or solution trace is being retrieved to guide the adaptation process. Adaptation-guided retrieval can be done through direct input from the designer looking for similar adaptation episodes, or through automated detection of these adaptation episodes or sequences.

4.6.3 Adaptation of Sol-Traces (adaptation toolkit)

Once a Sol-Trace has been retrieved, it must be adapted to fit the new problem specifications. Several mechanisms for adaptation are used:

1. Parametric adjustments of Sol-Traces
2. Replay of Sol-Traces with new parameters or in new problem situation.
3. Application of operations that have been previously encoded.

The Sol-Trace contains attributes and elements that are abstracted from the case. These attributes and elements are part of a hierarchy of alternatives. The Sol-Trace is a path through these alternatives with the

reasoning behind the choices. It is possible to traverse the hierarchies and choose alternatives that match the new problem specifications. CBR can be applied to this process recursively until a solution is composed. The method of putting the resulted choices together can be as simple as substitutions or more involved as propagation of the changes in replaying the trace.

The replay of Sol-Traces means that the Sol-Traces is being rerun in under the new problem specifications or characteristics and each decision, alternative, and method included should be reapplied in these new circumstances. The output should be a new solution that best fit the new problem. This ultimate replay is not realistic and feasible for architectural solution because of the numerous decisions, alternatives, and methods involved in the process. This approach limits this solution space by formalizing the contents and limiting the operations. In architectural design, replay cannot be envisioned easily given the exploratory nature of the design process, e.g. what if during replay a new direction of solutions emerged to be more interesting than the original course. Therefore, the notion of replay must be adjusted to fit the architectural requirements. Firstly, the research proposes different levels of replay in terms of automation. One level is where designers themselves anticipate the result of a replay and build the components of this replay, but they cannot visualize nor predict the results. Another level would be where designers choose certain operations and arguments that are within Sol-Traces and ask the system to apply them. The highest level would be to delegate the choice of these operations and the assigning of their arguments to the system. The research proposes an iterative and gradual development of these replays starting from the lower and moving to higher levels.

The adaptation of Sol-Traces provides a mechanism for exploring different venues of form composition. In creative design, the production of solutions cannot be separated from formulation of the problem, so exploring and expanding the design problem specifications and searching for characteristics of solutions should be integrated. The objective is to provide an unconstrained environment for using any case, characteristics, attribute, or method that is beneficial at any point. The goal is to avoid forcing designers to describe what and how they would like to explore, or restricting the creative flow of ideas. As Mitchell concludes:

> …any successful attempt to describe the mechanics of some ‘creative’ design activity will have the immediate effect of redefining that activity as ‘non-creative’
>
> [Mitchell, 1990, p.25]

4.6.4 Design composition through Sol-Traces

Through the adaptation/generation process, the resulted patterns of Sol-Traces depend on many factors. These factors are based on the contents of these traces, the type of adaptation used, and the CBR functions related to Sol-Traces. There are two main groups of factors:

A. **Mechanical factors** (operators): These factors are based on the actual types of operators, such as stretching, trimming, or splitting etc., used on the Sol-Trace or part of it. The resulting Sol-Trace would be the sum of interactions of the behaviors of components under certain priorities. For example, while stretching a building, the result changes as the stretch increases; at one point, it is a rectangular but at a further point, a new shape will emerge.

This behavior allows for unexpected discoveries of emergent forms. The process is set up to be dependent on the contents of the selection window and the collective behavior of these contents.

B. **CBR factors:** A change introduced by a designer in the adaptation process, could trigger a search and retrieval of another case that can guide the result of such change. This is similar to adaptation-guided retrieval. The retrieved case should have closer resemblance to the result the designer envisioned by triggering this particular change. Also, the retrieved case could switch the adaptation to other unpredictable twists during the form development.

To achieve this behavior, a common ground of elements and operations must be developed. Sol-Traces representation can provide, to some extent, satisfactory common categories of these elements and operations. Cases can be connected to the elements of Sol-Traces. The adaptation operations must be encoded in a standard way to allow matching between cases. Thus, while adapting, a search could be launched with a set of elements and a sequence of operations to be matched in other Sol-Traces. When this search is initiated is dependent on designer preference. It should not interrupt a flow of adaptation operations that a designer is deeply involved in. This approach is similar to the adaptation guided retrieval and the adaptation case approaches, except that it is applied under relaxed criteria. In general, Sol-Traces patterns can be guided by the following:

- The internal structure(s) and their dependencies including elements and operations
- Designer overriding – or introducing new relations etc.
- Cases that guide the behavior, either automatically retrieved (triggered) or manually retrieved by the designer (designer's choice)

The implication of this Sol-Traces behavior is that different shape components and operators, or shape manipulation operations, should evoke specific sets of shape transformation operators. This resembles the concept of type – if shape or form typologies can be recalled, for example a square shape as a type implies (or evokes) the use of particular operators and transformation strategies convenient to the aspects of the square shape. This concept is also utilized in the indexing and retrieval vocabulary of forms used in this research as established connections.

4.7 Integrated design process model

One of the objectives of this research is to create an integrated model for design composition using the Sol-Traces representation. The research uses s "design space" as the work space for exploring, representing, and reusing design cases. The design space will be used to achieve the following functions:

- To save the design sessions, including all the resources such as cases, traces, or case components, and possibly the designer's comments, for later continuation.
- To be used as a medium for extracting design form composition knowledge, building Sol-Traces, and reusing and adapting cases or traces.
- To allow more than one Sol-Traces to be retrieved at different points and to keep track of them.

- To allow effective navigation of the steps of the process and to keep track of resources used in the design session.
- To allow interaction between the designer and the system in developing and reusing Sol-Traces. The interaction that is proposed in this research is a shared interaction model; the system is used as a partner in the process. This is reflected in the proposed representations of Sol-Traces and in the proposed process for developing, indexing, retrieving, and adapting of these Sol-Traces.
- The integrated model includes the templates for building Sol-Traces.

5. Implementation

5.1 Object-Oriented Software Engineering (OOSE)

Object-oriented software development methodology will be used for the prototypical system that implements the intended functionality. The development process will start with identifying useful scenarios and use cases. These use cases will be used to guide the development and to be used as yardsticks to measure the resulting outcomes. The process will continue through developing the object model and the dynamic (behavior) model. An OO programming language is required and a graphical editor will be chosen. Details of the development process are presented in the following sections.

5.2 Requirements Modeling

In OO, scenarios and experiments are developed to obtain insights about the applicability of this approach. They are used to sketch out preliminary requirements of design composition assistance system and to examine the possibility and the feasibility of developing such system.

5.2.1 Experiments

A set of experiments were conducted to simulate, explore, and refine the proposed approach in different situation. These experiments helped to shape a feasible approach for the research problem and in giving insights of the proposed design composition assistance.

Experiment 1: Development of some form interpretations using the proposed language and the suggested categories. This experiment was at the start of formulating the research approach. Several buildings floor plans were abstracted using shapes and operations. The objective was to demonstrate that the language is capable of representing the form characteristics of these buildings. Clark's book, *Design Precedents*, was used as a source for floor plans of different buildings [Clark, 1986].

Experiment 2: Abstraction of Sol-Traces from F.L. Wright drawings for Usonian houses. In this experiment, a set of houses design by F.L.W. abstracted and a set of rules for developing these houses was extracted. The purpose was to build Sol-Traces from these abstractions and to achieve particular design composition goals. The experiment tested the Sol-Traces approach in regenerating or replaying under new different problem specifications such as providing four bedrooms instead of three bedrooms house. Some of the questions addressed are: how to add one or more bedrooms, how to applying different

shapes for the composition, and how to adapt the plan to fit different circumstances [Sergeant, 1975, McCarter et al., 1991.].

Experiment 3: Creation of more elaborated interpretations of design forms and design interactions. In these experiments, CBR functionality was applied to simulate a system to retrieve related cases and Sol-Traces. The indexing of these traces in the case base was also examined. Advanced replay and auto-generation of Sol-Traces as solutions were simulated.

5.2.2 Scenarios and use cases

A set of representative scenarios will be developed. Using these scenarios, use cases will be constructed and used to identify the objects and their classes.

Examples of possible scenarios:

- A designer uses Sol-Traces to compose a design form
- A designer uses Sol-Traces to explore possible design solution characteristics
- A designer adapts Sol-Traces to fit the new design problem
- A designer allows the system to suggest a trace that can guide the adaptation process
- A designer query the system for characteristics of the form (solution) for a design problem

In the sample scenario outlined below, a designer uses Sol-Traces to compose a new form for a school design:

1. Designer works on a design of a school (type of school, number of students, classrooms, facilities are outlined in the program)
2. Designer explores characteristics of the design solution: the form
3. Designer, using characteristics provided by the system, inputs the desired characteristics
4. The system retrieves a set of cases, in the form of Sol-Traces, that match these characteristics
5. Designer explores one of these Sol-Traces, included within the case, which can also be viewed and explored
6. Designer switches to another case within the set and explores its Sol-Traces.
7. In the Sol-Traces, the designer views the functional abstraction of the trace, compares it with what is required in the new design problem, and decides to add another multi-purpose room, as required by the program
8. Designer also explores the form abstraction. He decides the form is worth adapting and exploring
9. Designer replays the Sol-trace, as this function was available for this trace. The replay process accommodates the new form/function changes
10. The result of the replay is displayed to the designer: a repeated shape of the multi-purpose room in the original design.

11. The designer further refines the form through enlarging another space to maintain a gradation pattern of the shape of the multi-purpose room and confirms this as gradation to the system
12. The system reacts to the change and displays a set of building forms with gradation close to the current solution
13. Based on one form, the designer decides to enhance the gradation pattern and to end the design session with this form for further evaluation

5.3 Implementation

Through the use cases, objects will be identified and objects model(s) will be developed. Also, interactions between identified objects will be detected from the use cases. Identified objects will be organized in the static object model. Classes of objects can be created along with their attributes and behavior. A programming language that supports object oriented development such as Java will be used. A suitable development environment will be selected such as Jbuilder or J++. The choice of graphical editor can range from simple Java 2D editor for simple solution traces, to commercial CAD software such as AutoCAD, for more graphically elaborated solution traces. For this research prototype, a simple editor should prove sufficient to demonstrate the intended functionality. The user interface will also be built within a Java environment.

One advantage of OO technology is the representation of the proposed typologies (function and form). Typologies require a representation that allows for multiple classifications and an inheritance mechanism for attributes and characteristics. OO languages, such as Java, provide constructs for such representation and have a robust inheritance mechanism of classes.

CBR functionality will be provided by using a data base system in the background. Either an object oriented data base or an object relational data base can be used in this prototypical system [Harrington 2000, Stonebraker 1999]. The use of object oriented technology can enhance the retrieval through tying the data base indexing structures with similarity aspects. This technique builds on Kolodner's structured memory model of retrieval [Kolodner, 1993; Schank, 1982]. Scaling up the system might require a dedicated case based engine, or, if suitable, the use of commercially available case based development environments.

6. Research tasks, phases, and schedule

There are three main categories of research tasks:

- Tasks to explore and to investigate the research approach / thesis. These tasks cover the theoretical and experimental analysis that will lead to substantial answers to the research problem and support the hypothesis.
- Tasks related to the development of the proof-of-concept application.
- Tasks related to the documentation of the research and writing the dissertation.

The following are the core phases to the proposed research:

1. Developing the design composition methodology using Sol-Traces
2. Preparing the cases and a list of templates to input Sol-Traces in the case base
3. Building the case base (indexing – similarity measures – retrieval)
4. Developing the application requirements, objects, and behavior models
5. Developing the system architecture and implementation
6. Dissertation document preparation

1. Develop design composition methodology using Sol-Traces

Tasks:

1. To compose a systematic method/details for each component of the Sol-Traces used in design compositions
2. To develop demonstration examples – in details
3. To develop scenarios and use cases

Deliverables:

1. Written document explaining the methodology and the examples
2. Written enumeration of all Sol-Traces components (the developed language)
3. Scenarios / Use cases

2. Prepare cases and a list of templates to input Sol-Traces

Tasks:

1. To develop a list/chart of details of templates
2. To write the scenarios / use cases / interaction of using and customizing templates
3. To describe a set of cases to be used

Deliverables:

1. Written templates details / use cases
2. A set of cases

3. Build the case base (indexing - similarity matching – retrieval)

Tasks:

1. To develop indexing vocabulary
2. To develop similarity measures
3. To develop retrieval mechanisms and timing
4. To build the sample case base
5. To select data base to simulate the case base functionality

Deliverables:

1. Written document explaining the case base and the CBR system
2. Examples of simulated indexing / similarity matching /retrieval

4. Develop the application requirements, objects, and behavior models

Tasks:

1. To create a list of requirements
2. To develop object model
3. To create the interaction diagrams
4. To select the development environment and language
5. To select the graphical editor
6. To develop the connection between coding and graphical editor (if separate)

Deliverables:

1. Code for objects / documented an object model development
2. Documented interaction diagrams
3. Examples of cases presented in the graphical editor

5. Develop the system architecture and implementation

Tasks:

1. To develop system components and system architecture (interface, graphical editor, CBR engine, Sol-Traces engine)
2. To implement the required functionality (CBR + Sol-Traces)
3. To run examples to present the concept

Deliverables:

1. Prototype of a CBR system that supports Sol-Traces concept
2. A set of results (printed) of the functionality / behavior of the system

6. Dissertation document preparation

Tasks:

1. To outline the list of chapters
2. To document outcomes of previous phases
3. To write the conclusions, contributions, and future research

Deliverables:

1. Thesis chapters – at scheduled times
2. The complete dissertation

6.1 Research schedule

	Dec'03	Jan'04	Feb'04	Mar'04	Apr'04	May'04	Jun'04	Jul'4	Aug'04
Composition methodology (Sol-Traces)									
Cases and templates for Sol-Traces									
Develop case base									
Application (requirement, objects, behavior)									
System architecture and implementation									
Documentation and dissertation									

7. Conclusions and contributions

7.1 Conclusions

This thesis proposal presents a process model for design composition and a representation for the formal characteristics of design solution composition. The design process model is based on adapting previous solutions to generate new designs in a generative way using derivational analogy. The model builds on the cognitive aspects of the design process that can be applied either in traditional practice or using the computational media. The model relies on a proposed representation of design composition using a visual language. Solution traces are used within a CBR methodology in a computer system for design composition assistance. The research proposes a new technique for computer assisted design composition. A prototype system utilizing this approach is proposed as a proof of concept. The envisioned system provides description of crucial parts of the conceptual model. The proposed system will implement the crucial aspects of the approach; the Sol-Traces formalism and the process of design composition.

Building Sol-Traces and representing the reasoning behind them is a challenging process since both domain knowledge and problem solving knowledge have to be represented. The following are suggestions to avoid such difficulty:

1. Limiting the solution path to the tracing and remembering of alternatives, either the used ones or the ones available to the designer in that particular setting. This can be done without recording the reasoning process; in other words, without providing the reasons behind choosing one alternative or even behind having this set of alternatives available. This semi-reasoning can be complemented with the judgment of the designer in the new problem.

2. Limiting the scope of the problems/solutions pair, it is necessary to provide clear and simple choices of reasoning. So the problem and expected solutions have to be precisely defined, and reasoning choices have to be enumerated for the direct selection. The trace in this case must be the encountered route between these reasoning steps.

3. Limiting the number of and building elaborative traces that capable of yielding different solutions. The reasoning would be the choice of the trace and the mechanisms within the trace to accommodate the new problem circumstances. This could be achieved through standardizing the traces and how they will be followed or building virtual traces with standard alternatives and methods within them.

7.2 Contributions

The expected contributions of the proposed research are in three main research areas: a new approach for the design composition, computer assistance for early phases of the design process, particularly for design composition, and a new approach for generative CBR in design using derivational adaptation.

7.2.1 New approach for design composition

In this research, design composition is represented using a visual language of unit forms and transformations. This representation enables the emerging design concept to change and elucidates the conceptual structure of the design composition. This representation enables designers to see constructs and analogies of previously used or encountered concepts. This approach of abstracting the design composition contributes to the knowledge of both design composition and computer-aided design by allow: using practical and low level operations to build innovative design solutions, providing designers with tool-box for creating creative design solutions, and using constructs of conceptual design that can be implemented computationally.

7.2.2 Computer assistance for design composition

Attempt at representing the human design process computationally has proved to have many limitations. Therefore, taking advantages of what machines are good at and incorporating this within the design process has greater potential. The computational problems are much more difficult in the early phases of the process. For computers to assist in this phase, new abstractions and new constructs that are understandable by both designers and computers are required. This research presents an

abstraction/representation process that result in such constructs. This contribution is applicable to the design in architecture and other design fields in general.

7.2.3 Derivational analogy for adaptation and generative CBR

Representing design path/history (components of design rationale) is one of the promising approaches in computational design. Derivational analogy is at the heart of this approach. Combining CBR methodology and derivational analogy provides benefits from both methodologies. Applying this to the architectural field provide successful examples of strategies for design computation is among the contributions of this research.

The proposed generative approach is of significant importance to the architectural design, where direct reusing or adapting solutions are not favored. Recording design history and design path provide mechanisms for generating desired solutions. The approach attempts to formalize this path/history and utilizes it in computational design. Form-function association is one of the characteristics recorded in the design path/history. Having a system that can capture such association cannot be underestimated. The approach provides for this design knowledge to be captured, represented, and reused in many levels of design composition. These associations can be used in searching for designs, exploring design aspects, or composing design solutions.

In CBR, adaptation is considered one of the most difficult tasks. The approach provides an adapted model of CBR, combined with derivational analogy and provides different approach for the adaptation problem. Mainly, based on derivational analogy model, adaptation is intertwined within the functionality of CBR and built within case representation. In this way, the approach provides a new understanding of computational methodologies such CBR and derivational analogy and how to adapt them to the particular reasoning in the early phases of the architectural design process.

7.2.4 Future work

The proposed design composition assistance provides the basics of the design process model. Further extensions, automations, and support can be achieved in future research projects. In the reuse and adaptation aspects, two points of future extensions are envisioned:

1. Automatic composition of solutions using the recorded associations
2. Automated building and advanced replays of Sol-Traces
3. Automated recognition of the elements of the visual language

8. References

Aamodt, A. and Plaza, E. (1994). "Case-based reasoning: Foundational issues, methodological variations and system approaches." *AI communications*, Vol. 7 No. 1, pp. 39-59.

Alexander, Christopher, (1977). *A Pattern Language*. Harvard University Press; Cambridge, MA.

Baker, G.H. (1993), *Design Strategies in Architecture: and approach to the analysis of form*, Van Nostrand Reinhold, UK.

Bhatta, S., Goel, A., and Prabhakar, S. (1994) "Innovation in Analogical Design: A Model Based Approach," *Proc. Third Int'l Conf. AI in Design*, Kluwer, pp. 57-74.

Bonnardel, N. and R. Megali (1998). "Analogies in Design Activities: A Study of the Evocation of Intra- and Interdomain Sources." In K. Holyoak, D. Gentner, and B. Kokinov, *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences*, Bulgaria, pp.336-343.

Borner, K. (2002), *Visual Interfaces to Digital Libraries*, Springer-Verlag, Germany.

Borner, K. (1998), "CBR for design." In *Case-Based Reasoning Technology From Foundations to Applications,* Lecture Notes in AI, Vol 1400, M. Lenz, B. Bartsch-Sporl, H.D. Burkhard, and S. Wess. eds, Springer-Verlag, New York.

Bruton, D. (1997). "Grammars and Art." In *CAAD Futures 1997, Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures*, Munich, Germany, pp.71-82.

Carbonell, J.G. (1985). "Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition." Tech. Report, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

Ching, F. (1979), *Architecture: Form, Space and Order*, Van Nostrand Reinhold, New York.

Chiu, M-L. and Shih, S-G. (1997). "Analogical Reasoning and Case Adaptation in Architectural Design: Computers vs. Human Designers." In *CAAD Futures 1997, Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures*, Munich, Germany, pp.787-800.

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

Cross, N., Christiaans, H., and Dorst, K. eds., (1996), *Analyzing Design Activity*, J. Wiley & Sons.

Cunningham P., Finn D., and Slattery S. (1994). "Knowledge Engineering Requirements in Derivational Analogy." In *Topics in Case-Based Reasoning, Lecture Notes in Artificial Intelligence*, S. Wess, K-D Althoff, M.M. Richter eds., Springer Verlag, pp234-245.

Domeshek, E.A., Zimring, C.M. and Kolodner, J.L. (1994). "Scaling up is hard to do: Experiences in preparing a case-based design aid prototype for field trial." *ASCE. American Society of Civil Engineers '94*. pp 430 - 437.

Domeshek, E. and Kolodner, J.L. (1992). "A case-based design aid for Architecture." *Artificial Intelligence in Design '92*, The Netherlands, pp. 497-516.

Downing, F. (2000). *Remembrance and the Design of Place*, Texas A&M Press, College Station TX, pp. 144.

Falkenhainer, B., Forbus, K.D., & Gentner, D. (1989). "The structure mapping engine: Algorithm and examples." *Artificial Intelligence*, Vol 41, pp.1-63.

Finger, S. (1998). "Design Reuse and Design Research." Design Reuse, Edited by S. Sivaloganathan and T.M.M. Shahin, *Proceedings of Engineering Design Conference '98*, Professional Engineering Publishing Ltd., London, UK, pp.3-9.

Flemming, U. (1987). "More Than the Sum of Parts: The Grammar of Queen Anne House." *Environment and Planning B. Planning and Design*, Vol 14,323-350.

Flemming, U. and Woodbury, R. (1995) Software environment to support early phases in building design: Overview. *Journal of Architectural Engineering*, Vol 4, No 1, pp 147-152.

Flemming, U., (1990). "Syntactic Structures in Architecture: Teaching Composition with Computer Assistance." In Mitchell, W. and McCullough, M. eds. *The Electronic Design Studio*, M.I.T. Press, Cambridge, MA, pp.31-48.

Flemming, U. (1994). "Artificial Intelligence and Design: A Mid-Term Review." In G. Carrara and Y.E. Kalay, eds., *Knowledge-Based Computer-Aided Architectural Design*, Elsevier, Netherlands, pp. 1-24.

Flemming, U. (1994a). " Case-Based design in the SEED System." in G. Carrara and Y. E. Kalay, eds., *Knowledge-Based Computer-Aided Architectural Design*, Elsevier, Netherlands, pp. 69-91.

Frazer, J. (1995). *An Evolutionary Architecture*, Architectural Association, London.

Gargus, J. (1994). *Ideas of Order: A Formal Approach to Architecture*, Kendall/Hunt Publishing Co., Iowa.

Gasakin, H. and G. Goldschmidt (1999). "Expertise and the use of visual analogy: implications for design education." *Design Studies* Vol. 20, No. 2, pp. 153-175.

Gentner, D., (1983), "Structure mapping: a Theoretical Framework for Analogy." *Cognitive Science*, Vol 7, No 2, pp. 155-170.

Gentner, D. (1988). "Structure-Mapping: A Theoretical Framework for Analogy." In Collins, Allan and Smith, Edward E. eds, *Readings in Cognitive Science A Perspective from Psychology and Artificial Intelligence*, Morgan Kaufmann Publishers, pp. 303- 310.

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

Goldschmidt, G. (1995). "Visual Displays for Design: Imagery, Analogy and Databases of Visual Images." In Koutamanis A. ed., *Visual Databases in Architecture*, Avebury, pp.53-74.

Hampton, J. A (1998). "The Role of Similarity in How We Categorize The World." In Holyoak, K., Gentner, D., and Kokinov, B., *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences*, Bulgaria, pp.19-30.

Harrington, J.L. (2000), *Object-Oriented Database Design Cleary Explained*, Academic Press, CA.

Hovestadt, L. and Hovestadt, V. (1997). "Armilla 5- Supporting Design, Construction and Management of Complex Buildings." *In CAAD Futures 1997, Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures*, Munich, Germany, pp.135-150.

Jakimowicz, A., Barrallo, J., and Gueded, E.M. (1997). "Spatial Computer Abstraction: From Intuition to Genetic Algorithms." In *CAAD Futures 1997, Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures*, Munich, Germany, pp.917-926.

Jones, J.C., (1966). "Design Methods Reviewed." In: S.A. Gregory (ed.), *The Design Methods*, London, Butterworth, and New York, Plenuum Press, pp. 295-309.

Juris, I.J. (1993). "Representation and Management Issues for Case-Based Reasoning Systems." Department of Computer Science, University of Toronto, Ontario, Canada.

Kedar-Cabelli, S. (1988). "Analogy-From a Unified Perspective." In Helman, D.H., ed., *Analogical Reasoning: Perspective of Artificial Intelligence, Cognitive Science, and Philosophy*, Kluwer Academic publishers, Netherlands, pp.65-103.

Knight, T.W., (1999). "Shape grammars applications in architectural design, education, and practice." *Report for the NSF/MIT*, Department of architecture, MIT, Cambridge, MA.

Kolarevic, B. (1997). "Regulating Lines and geometric Relations as a Framework for Exploring Shape, Dimension and Geometric Organization in Design." In *CAAD Futures 1997, Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures,* Munich, Germany, pp.163-170.

Kolodner, J.L. (1993). *Case-based reasoning*, Morgan Kaufmann, New York.

Kolodner, J.L. (1996). "Making the Implicit Explicit: Clarifying the Principles of Case-Based Reasoning." *Case-Based Reasoning experiences, Lessons, and Future Directions*, D.B. Leake, ed., MIT Press, Cambridge, MA, pp.349-370.

Konar, A. (2000), *Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain*, CRC Press, Florida.

Krier, R. (1988), *Architectural Composition*, Rizzoli, New York.

Kuhn, C. and Herzog, M. (1994) "Modeling the Representation of Architectural Design Cases." *In Automation Based Creative Design*, edited by Tzonis A. and I. White, Elsevier, Netherlands, pp.69-82.

Lauer, D.A. (1985). *Design Basics*, Second Edition, CBS College Publishing, New York.

Leake, D.B. (1996). ed. *Case-Based Reasoning Experiences, Lessons, and Future Directions*, MIT Press, Cambridge, MA.

Lenz, M., B. Bartsch-Sporl, H.D. Burkhard, and S. Wess (1998). eds. *Case-Based Reasoning Technology From Foundations to Applications,* Lecture Notes in AI, Vol 1400, Springer-Verlag, New York.

Leupen, B., C. Grafe, N. Kornig, M. Lampe and P. Zeeuw (1997). *Design and Analysis*, Van Nostrand Reinhold, New York.

McCarter, R. (1991) ed., *Frank Lloyd Wright: A Primer on Architectural Principles*, Princeton Architectural Press. NY.

Maher, M.L. and Zhang, D.M. (1993). "CADSYN: A case-based design process model." *AIEDAM*, Vol 7, No 2, pp. 97-110.

Mitchell, W.J. (1994). "Artifact Grammars and Architectural Invention." In *Automation Based Creative Design*, ed Tzonis A. and I. White, Elsevier, Netherlands, pp.139-159.

Mitchell, W.J. (2002), "E-Bodies, E-Building, E-Cities.". In Leach, Neil, ed. *Designing for a Digital World*, Wiley-Academy, Italy.

Mitchell, W.J., (1977). *Computer Aided Architectural Design*, New York.

Norberg-Schulz, C. (2000). *Architecture: Presence, Language and Place*, Skira Editore, Milan, Italy.

Novah, M. (2001) "Liquid~, Trans~, Invisible~: The Ascent and Speciation of the Digital in Architecture. A Story". In Schmal, P., *Digital Real*, Birkhauser Pub. Germany, pp. 214 - 247.

Oxman, R. and Oxman R. (1994). "Remembrance of Things Past: Design Precedents in Libraries." In *Automation Based Creative Design*, Tzonis A. and I. White, eds Elsevier, Netherlands, pp. 55 - 68.

Oxman, R. and Oxman R. (1996). "The Computability of Architectural Knowledge." In *The Electronic Design Studio*, McCullough M., Mitchell W.J., and Purcell P. eds., The MIT Press, Cambridge, MA, pp.171-185.

Oxman, R. and Oxman R (1994). "Case-Based Design: Cognitive Models for Case Libraries." In G. Carrara and Y.E. Kalay, eds., *Knowledge-Based Computer-Aided Architectural Design*, Elsevier, pp 45-68.

Oxman, R., Radford, A., Oxman, R. (1987). *The Language of Architectural Plans*, Royal Australian Institute, Australia.

Pu, P. (1993). "Introduction: Issues in Case-Based Design Systems." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Academic Press Ltd., Vol. 7, No 2, pp. 79-85.

Quin L., and Gero, J. (1992) "A Design Support System Using Analogy," *Proc. Second Int'l Conf. AI in Design*, Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 795 - 813.

Raduma, P. (1999). "Designing a CBR Representation Scheme for Architectural Design Reuse." Arpakannus 1/99, *Newsletter of the Finnish Artificial Intelligence Society, Special Issues on the IJCAI'99* affiliate event Networks'99, Espoo, Finland August 8-19.

Raduma, P. (2000). *Case Based-Reasoning in Knowledge-Based CAAD: Modeling Case* Representation for Architectural Design Reuse. Ph.D. dissertation. Dept. of Architecture, Helsinki University of Technology, Espoo, Finland.

Rivard, H. and Fenves S.J. (2000) "A Representation for Conceptual Design of Buildings." *Journal of Computing in Civil Engineering*, Vol. 14 No.3, pp. 151-159.

Rowe, (1987), *Design Thinking*. MIT Press, Cambridge, MA.

Schaaf, J.W. and Voss A. (1995). "Retrieval of Similar Layouts in FABEL using AspecT." In *CAAD Futures '95, Proceedings of the Fifth Int'l Conf. on Computer-Aided Architectural Design Futures*, Singapore, School of Architecture, National University of Singapore.

Schank, R., (1982). *Dynamic Memory*, Cambridge University Press, Cambridge UK.

Schmitt, G. (1994). "Case-Based design and Creativity." In *Automation Based Creative Design*, edited by Tzonis A. & I. White, Elsevier, Netherlands, pp.41-53.

Schmitt, G. (1995). "Architectura cum machina-interaction with architectural cases in a virtual design environment." In *Visual Databases in Architecture*, A Koutamanis. ed., Avebury, pp.113-128.

Sergeant, J. (1975). Frank *Lloyd Wright's Usonian Houses: The case for organic architecture*, Whitney Library of Design, NY.

Sklar, H.F. (1995). "Opening DOORS: Online access to design resources." In *Visual Databases in Architecture*, A Koutamanis. ed., Avebury, pp.53-74.

Smyth, B. and Keane, M.T (1996). "Design a la Deja Vu: Reducing the Adaptation Overhead." In *Case-Based Reasoning experiences, Lessons, and Future Directions*, Leake, D.B., ed., MIT Press, Cambridge, MA, pp.151-166.

Stevens, G. (1990), *The reasoning architect: mathematics and science in design*, International Editions, Architecture Series, McGraw-Hill, New York.

Stonebraker, M., P. Brown and D. Moore, (1999) *Object-Relational DBMSs tracking the next great wave*, Morgan Kaufmann.

Thagard, P. (1988). "Dimensions of Analogy." In *Analogical Reasoning: Perspective of Artificial Intelligence, Cognitive Science, and Philosophy,* Helman, D.H., ed., Kluwer Academic Publishers, Netherlands, pp.105-124.

Turner, M. (1988). "Categories and Analogy." In *Analogical Reasoning: Perspective of Artificial Intelligence, Cognitive Science, and Philosophy,* Helman, D.H., ed., Kluwer Academic Publishers, Netherlands, pp.3-24.

van Leusen, M. (1995). "Type representations in case-based design." In Koutamanis A. ed., *Visual Databases in Architecture*, Avebury, pp.75-88.

Watson, I. and S. Perera (1997). "Case-Based Design: A Review and Analysis of Building Design Applications," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Cambridge University Press, Vol. 11, pp. 59-87.

Wong, W. (1988). *Principles of Two-Dimensional Form*, Van Nostrand Reinhold Inc., New York.

Wong, W. (1993). *Principles of Form and Design*, Van Nostrand Reinhold, New York.