CBR as a Framework for Design:

Augmenting CBR with other AI techniques

Mary Lou Maher

Department of Architectural and Design Science
University of Sydney
Sydney, NSW 2006
AUSTRALIA
mary@arch.usyd.edu.au

Abstract

Design is an activity in which the role of experience plays a larger part in the generation of alternative designs than theoretical or formal knowledge. This has lead to increasing interest in CBR as a way of assisting and/or automating portions of the design process. However, there are major areas in the CBR paradigm that do not address the needs of practical CBR. Here CBR is presented as a framework for design, with other AI techniques supporting different aspects of CBR. Specifically, knowledge discovery is used to assist in the development of CBR knowledge and genetic algorithms in the generation of design solutions.

Introduction

Design is characterised as a creative act in which a new product emerges from the knowledge of previous products and experience in response to a set of needs. Associated with the process of design is:

- the importance of experience, where novice designers rely on the experience of others as embodied in previous designs and expert designers rely on the richness of their own experience;
- a lack of generalisations, where there are few rules for producing designs that do not have significant exceptions;
- a lack of domain theories for synthesis, since the well known aspects of a particular domain typically cover analysis of designs as a deductive process; and
- the role of previous designs as stories, where designers are more comfortable telling stories about their experiences rather than providing domain theories or heuristic rules.

In addition to this characterisation of the design process as being creative, informal, and experience-oriented, the design profession rarely accounts for the design *process*, but rather records the design experience as documented *projects*. Therefore, we consider CBR as an overall framework for intelligent design support and look at how other AI techniques fit in that framework.

Case-Based Reasoning (CBR) is a reasoning paradigm used in Artificial Intelligence that uses knowledge in the form of previous experiences as the basis for solving new problems. The main "reasoning cycle" employed by a CBR framework for design is shown in Figure 1.

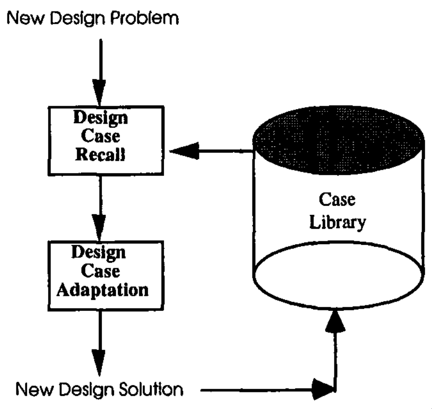

Figure 1. A CBR framework for design

A new design problem serves as an index to a case library. A set of design cases are recalled and become the basis for design case adaptation. A new design solution is then included in the case library, allowing the system to learn as it experiences new situations. This is a simplistic framework, with some important characteristics: a new design problem is not well defined, implying that the cases that are recalled may not be a good fit; and that adaptation is required in all situations, since every design is unique. Since the problem is not well defined, there are issues related to the way in which designs are indexed and retrieved that are not well addressed by the CBR paradigm.

Design case adaptation raises issues related to the need for methods to change previous designs and to recognise the adequateness of new designs. Such methods are not inherent in the CBR paradigm itself.

The Need for Integrating CBR with other AI Techniques

CBR as an approach to artificial intelligence assumes a representation of experiential memory, without specifying how that memory is indexed or acquired into the CBR system. CBR as a problem solving paradigm assumes the minimum of two processes: recall and adaptation. The paradigm itself does not indicate what methods are used to achieve these. Each of these areas have the potential for integrating the concept of CBR with other problem solving paradigms and/or AI techniques. For example:

1. Acquiring memory and memory indices: AI techniques such as knowledge acquisition techniques, and machine learning techniques such as conceptual clustering are useful.
2. Recalling cases: machine learning techniques such as induction can be used to develop a indexing tree; and pattern matching and similarity measures can be used for selecting cases from case memory.
3. Adaptation: various problem solving paradigms such as constraint satisfaction, heuristic search, genetic algorithms can be used depending on what kind of knowledge is available.

Based on our experience in developing case-based design systems (Maher et al 1995; Maher, 1997; Maher and Gómez, 1996) we recognise the following bottlenecks in developing case-based reasoning systems: representation and adaptation.

The representation and recall of previous designs requires a significant analysis and knowledge engineering of the design domain. Designers are happy to tell stories about their experience, but storing and recalling these stories in a formal system (or even an informal hypermedia system) is a difficult task. This is primarily due to the lack of formal models for representing design products that go beyond the documentation needed to manufacture or construct the product. The current documentation of design does not necessarily assist with the reuse of the design episodes, but rather provides information for the current situation.

Adaptation of previous designs requires both case and generalised knowledge. Design as a creative act implies that design adaptation is not a simple process, in fact it could be argued that adapting previous designs is the core of the design synthesis process. This aspect of CBR is not well developed and requires an analysis of the domain knowledge in order to achieve any kind of assistance or automation. The difficulty of this aspect of case-based design is reflected in the tendency for many CBR design systems to leave this part to the user, focussing of indexing and retrieval as the main contribution of CBR to design (see, for example, Maher and Pu, 1997). However, there are other AI paradigms that can support this aspect of CBR design.

In the next sections, the integration of CBR with other AI techniques is shown to address the adaptation bottleneck and the development of indexing and validation knowledge.

Adapting design cases using GAs

Adapting design cases requires techniques for changing and combining design cases, an inherently generative process, and techniques for evaluating the proposed design solution, an inherently analytical process. The types of knowledge-based methods applied to this part of the CBR cycle includes heuristic search and constraint satisfaction. One disadvantage that these methods have in common is the knowledge intensive nature of the methods for both generating proposed solutions and analysing a proposed solution.

As an alternative, we are exploring the use of Genetic Algorithms (GA's) (Maher and Gomez, 1996) to perform design case adaptation. GAs have several advantages with respect to most knowledge-based methods: they require less domain knowledge in order to search a space, while still producing "feasible" results; they are more flexible in that they are not limited to describing design cases using a pre-defined scheme with a fixed set and a fixed number of variables; and they inherently combine bits and pieces from several past experiences in order to solve a new problem, a capability that seems necessary for creative design.

Genetic Algorithms (GAs) provide an alternative to traditional search techniques by simulating mechanisms found in genetics. Three notions are borrowed from biological systems:

- the *phenotype*, which can be a living organism for biological systems or a design solution for design systems;
- the *genotype*, which is a way of representing or encoding the information which is used to produce the phenotype; and
- the *survival of the fittest*, which determines whether a genotype survives to reproduce.

A genetic algorithm starts with a population of potential design solutions, represented as genotypes. The partially-matching retrieved design cases provide the initial (seed) population for the genetic algorithm. Assuming a set of attribute-value pairs represent a design case, the set of attributes are equivalent to the genotype of a design. The values that are associated with the attributes in the description of a case represent the structural or behavioural embodiment of a specific design. The set of attribute-value pairs that make up a case description are collectively equivalent to the phenotype of a design.

The genetic algorithm operates on individual genotypes by randomly mating and mutating them, detecting any changes in the corresponding phenotypes, determining if any of the resulting phenotypes is good enough to represent a solution to the problem being solved, and repeating the process if not. Each phenotype is a potential solution to the design problem being solved. New potential solutions to the problem are generated through the combination of genotypes.

The first step required in the genetic algorithm is to convert the case representation of a design, which can be considered to be its phenotype, into a genotype representation of it. This genotype representation can be manipulated using the traditional genetic algorithm operators of crossover and mutation. The offspring genotypes resulting from this can be mapped back into phenotype form by assigning values to the genes in the genotypes. This results in new case-like descriptions that represent new potential solutions to the design problem being solved. The new phenotypes are evaluated to determine their feasibility as solutions to the design problem, and as potential "parents" for the next cycle (if any) of the genetic algorithm, and the process can be repeated, continuing until a satisfactory solution is found. Figure 2 illustrates the genetic algorithm for case adaptation.

This approach to design case adaptation provides a method for combining and changing design cases that requires little domain knowledge during the generative stage. The advantage of this approach is that it does not require formal domain knowledge for the generative aspects of design case adaptation. The genetic algorithm approach to design case adaptation requires the consideration of two issues: the evaluation of designs requires knowledge in the form of a fitness function and the representation of complex systems as genotypes requires a consideration of the structure of the phenotype/genotype representation. Although the GA provides a knowledge lean mechanism for generating possible designs, there are still the two bottlenecks: handcrafting the representation of cases as both genotypes and phenotypes, and the knowledge intensive task of evaluating alternative designs.

Learning recall and adaptation knowledge from case libraries

The development and implementation of design case libraries is an ongoing task. Often the domain is better understood after the case representation has been determined, and would be done differently if the project started again. In this section, AI techniques are used to learn indexing and adaptation knowledge from case libraries to ensure that the knowledge used during these tasks is based on the content of case memory, rather than on a preliminary understanding of the domain.

We have developed a multimedia case library of buildings that focus on structural design (Maher, 1997). The library is referred to as SAM, for its use in teaching Structures And Materials to undergraduate architecture students.

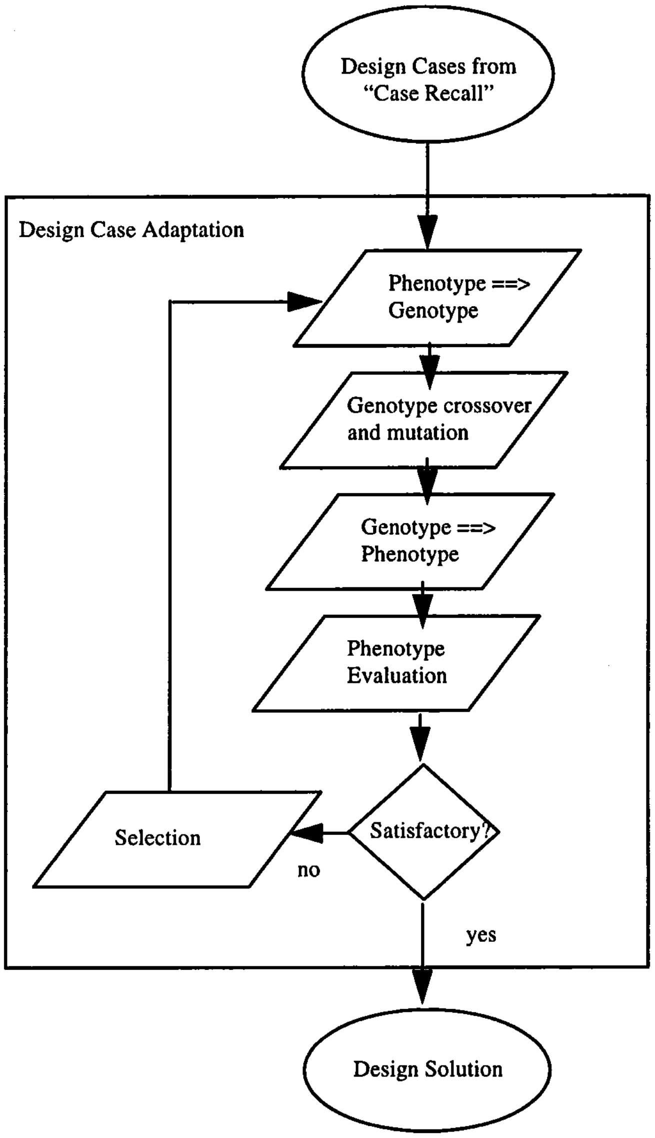

Figure 2. Design case adaptation using a GA

In developing SAM, we considered:

- the need to represent and manage *complex design* cases,
- the need to formalise a typically *informal body of knowledge* or experiences.

Design in any domain usually involves the development and understanding of complex systems. The complex representations needed to adequately capture a design case have introduced challenges to CBR systems. The CBR paradigm assumes that there is a concept of "a case", but in most design domains this concept is not simply "a case" but a complex set of experiences and decisions resulting in a complex system. Three approaches to addressing complexity are:

- a case is a *hierarchy of concepts*, or subcases
- a case is represented by *different views*
- a case is presented as *multimedia*

Case-based knowledge engineering inherited the methods used in the development of expert systems. The indexing scheme is generated either through the knowledge acquisition technique of *interviewing* the expert to identify the critical features, or through machine learning techniques to identify the most discriminating features by induction. The current approach to developing the adaptation knowledge is similar: the expert provides the rules and/or models of relevance. As a result, the shift in knowledge-base paradigm from expertise to experience identified new problematic issues related to the knowledge intensive indexing, retrieval, adaptation and maintenance of cases. However, this knowledge is not easily captured; the process is time-consuming, painstaking and complicated, requiring carefully developed questioning strategies, observational procedures and analysis methods. A logical consequence is the increasing attention of the case-base reasoning community towards machine learning algorithms (Hanney and Keane, 1996).

We are exploring the use of data *mining* and knowledge *discovery* (KD) techniques, recently developed for identifying useful implicit information coded in databases (Chen *et al.*, 1996), as a way of overcoming these difficulties. Viewing knowledge engineering as a discovery process means examining a data source for implicit information that one is unaware of prior to the discovery and recording this information in explicit form. This spans the entire spectrum from discovering information of which one has no knowledge to where one merely confirms a well known fact. Current KD methods are developed mainly for structure-valued data (Fayyad et al, 1996b; Williams and Huang, 1996).

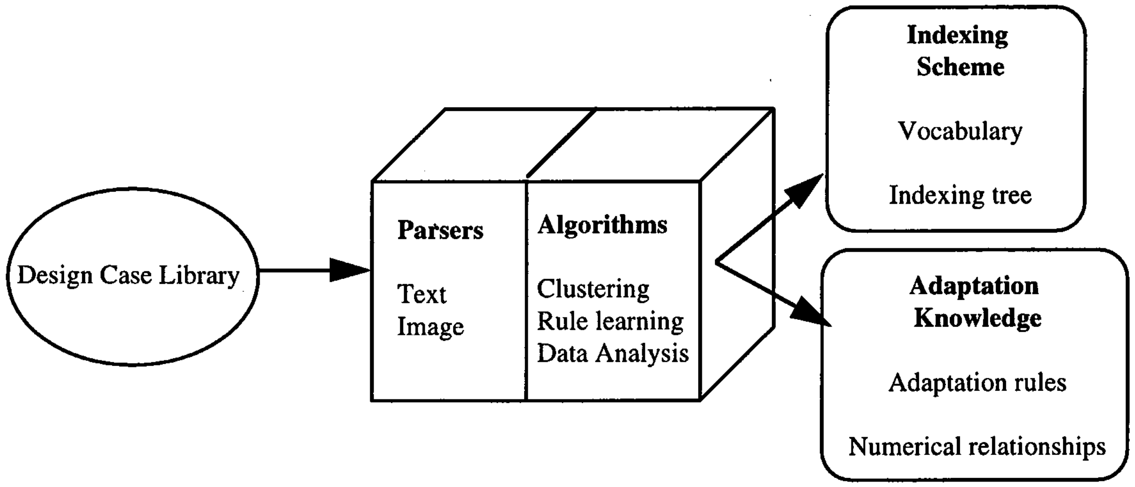

Figure 3. Overall process for knowledge discovery in design case libraries

Discovering implicit knowledge in case bases is substantially different to data mining (Maher and Simoff, 1997). The data organisation units in database mining are the *data columns*. Inside the case base the organisational unit is the *case*, which comprises a variety of data types and formats. Knowledge discovery then, in our use of the term, involves finding patterns in primarily unstructured, multimedia data.

Our model for knowledge discovery in design case libraries, as illustrated in Figure 3, takes a hypermedia library of cases, uses a number of knowledge discovery techniques in two phases to generate domain knowledge. We are using SAM as an existing multimedia representation of structural design cases and apply knowledge discovery techniques that can find patterns in the cases. Specifically, we are interested in finding patterns in the cases that can assist with *indexing* and *adapting* cases as a way of supporting the designer in being reminded of a previous experience and being informed when an adaptation lies outside the experience base.

We use two phases of knowledge discovery roughly corresponding to data-driven and expectation-driven approaches. In the first phase, the data-driven phase, a parser is applied to the case library to extract a set of relevant features. In the second phase, the features abstracted from the cases are used as input to various machine learning techniques to contribute to the indexing scheme and/or adaptation knowledge. Parsing cases to find relevant features produces a vocabulary and frequency of occurrence. We use two kinds of parsers, text parsers and image parsers, reflecting the multimedia nature of the design cases.

Once a set of features has been abstracted from the case description, there are several relevant techniques for transforming the features into knowledge patterns. The input, or training set, can be carefully constructed from the

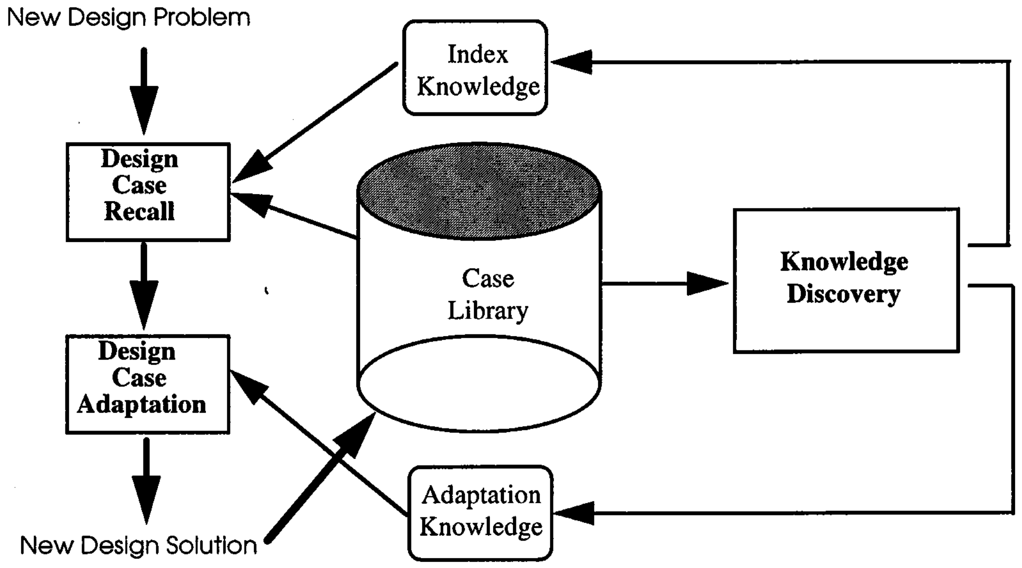

Figure 4. An enhanced model of CBR for design

features found during the parsing phase. Although the content is based on the case library, the format can be determined by the needs of the particular learning technique. The idea of expectation-driven discovery means that we start with an hypothesis of the knowledge to be discovered, such as an indexing scheme, and the learning algorithm modifies the knowledge. We have identified several classes of learning techniques that can be applied for knowledge discovery.

Conceptual clustering: These techniques are used in the machine learning community to synthesise generalised clusters from a training set of examples. In our project we use the vocabulary from the parsers above as the training data for conceptual clustering. The result of the application of these techniques will be a classifier tree that can partition the case library according to meaningful terms. Techniques for conceptual clustering are: CLUSTER (Michalski and Stepp, 1983), or more recent approaches specifically for clustering design concepts (Maher and Li, 1994).

Data analysis techniques: These techniques operate on the numerical valued parameters to find approximated linear equations that reflect the data in the cases. Candidate techniques for learning about numerical relationship are: EFD: Empirical Formula Discovery (Maher and Li, 1994), Interval Analysis (Voschinin *et al.*, 1993) and Function Discovery (Wu and Wang, 1991).

Rule learning techniques: These techniques are applied to the feature-valued pairs (numeric, boolean, symbolic). The idea is to compute the feature-differences that exist between cases and examine how these differences relate to differences in case solutions. Candidate techniques for rule learning are described in (Fayyad et al., 1996a; and Hanney and Keane, 1996).

Discovering knowledge from case data is still in its infancy. Part of the success of case-based reasoning as an interactive knowledge-based computing model is credited to its less demanding knowledge engineering and instant learning loop embedded in the model. However, the reality of the CBR approach is that knowledge engineering is still a critical and difficult part of the application. With the use of knowledge discovery techniques, we have an enhanced model of case-based reasoning, as shown in Figure 4, which includes AI techniques for knowledge formulation.

Summary

CBR as a framework for design puts the emphasis on the representation previous design cases and focuses reasoning on this representation. This is appropriate given the lack of formal knowledge about how designs are generated. However, the framework of CBR for design requires a consideration of other AI techniques as a means of representing and using design knowledge that is not easily or appropriately embedded in cases. Two of the AI techniques we consider here are: knowledge discovery for finding generalisations from case memory and genetic algorithms as a mechanism for case combination for the generation of new designs.

Acknowledgments

The ideas presented here are based on research done by Dr. Simeon Simoff in the areas of multimedia case based design and knowledge discovery, and by Andres Gomez de Silva Garza in the area of genetic algorithms for design case adaptation. This research has been supported by the Australian Research Council.

References

Chen, M-S., Han, J. and Yu, P.S. 1996. Data mining: an overview from a database perspective. *Knowledge and Data Engineering*, **8** (6):866-883.

Fayyad, U.M., Piatetsky-Shapiro, G. and Smyth, P. 1996b. From Data Mining to Knowledge Discovery in Databases, *AI Magazine*, **17** (3):37-54.

Hanney, K. and Keane, M.T. 1996. Learning adaptation rules from a case-base. In I. Smith, B. Faltings (eds), *Advances in Case-Based Reasoning*, 179-192, Heidelberg:Springer.

Maher, M.L. 1997. SAM: A multimedia case library of structural designs. In Y.T.Liu, J-Y. Tsou and J-H. Hou (eds), *CAADRIAí97*, 5-14, Taiwan:Huís Publisher Inc.

Maher, M.L., Balachandran, B., Zhang, D.M. 1995. *Case-Based Reasoning in Design*, New Jersey: Lawrence Erlbaum Associates.

Maher, M.L. and Li, H. 1994. Learning Design Concepts Using Machine Learning Techniques, *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, **8**(2):95-112.

Maher, M.L. and Gomez de Silva Garza, A. 1996. The Adaptation of Structural System Designs Using Genetic Algorithms. In Proceedings of the International Conference on Information Technology in Civil and Structural Engineering Design--Taking Stock and Future Directions, Glasgow, Scotland.

Maher, M.L. and Pu, P. eds. 1997. *Issues and Applications of Case-Based Reasoning in Design*, New Jersey :Lawrence Erlbaum Associates.

Maher, M.L. and Simoff, S. 1997. Knowledge discovery in multimedia design case bases. In B. Verma and X. Yao (eds), Proceedings ICCIMA'97, 6-11, Gold Coast:Griffith University.

Michalski, R.S. and Stepp, R. 1983. Learning From Observation: Conceptual Clustering, in Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (eds) *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.

Voschinin, A.P., Dyvak, N.P. and Simoff, S.J. 1993. Interval methods: theory and application in design of experiments, data analysis and fitting. In E. K. Letzky, (ed.), *Design of Experiments and Data Analysis: New Trends and Results*, 11-51, Moscow:Antal.

Williams, G. and Huang, Z. 1996. A Case Study in Knowledge Acquisition for Insurance Risk Assessment using a KDD Methodology, Data Mining Portfolio - TR DM 96023, CSIRO.

Wu, Y.-H. and Wang, S. 1991. Discovering functional relationships from observational data. In G. Piatetsky-Shapiro and W. J. Frawley (eds), *Knowledge Discovery in Databases*, 55-70, Cambridge: AAAI Press/ The MIT Press.