CASE-BASED REASONING IN DESIGN

Mary Lou Maher
M. Bala Balachandran
Dong Mei Zhang

CASE-BASED REASONING IN DESIGN

This page intentionally left blank

Case-Based Reasoning in Design

MARY LOU MAHER
University of Sydney, Australia

M. BALA BALACHANDRAN
University of Wollongong, Australia

DONG MEI ZHANG
CSIRO, Sydney Laboratory, Australia

Psychology Press
Taylor & Francis Group
NEW YORK AND LONDON

First Published 1995 by
Lawrence Erlbaum Associates, Inc.

Published 2014 by Psychology Press
711 Third Avenue, New York, NY 10017

and by Psychology Press
27 Church Road, Hove, East Sussex, BN3 2FA

Psychology Press is an imprint of the Taylor & Francis Group, an informa business

Library of Congress Cataloging-in-Publication Data

Maher, Mary Lou.
Case-based reasoning in design / Mary Lou Maher, M. Bala Balachandran, Dong Mei Zhang.
p. cm.
Includes bibliographical references and index.
1. Computer-aided design. 2. Engineering design—Case studies. I. Balachandran, M. (Muthaukumar), 1953–. II. Zhang, Dong Mei. III. Title.
TA174.M335 1995
620'.0042'0285—dc20 95-17344
CIP

ISBN 13: 978-0-805-81831-4 (hbk)
ISBN 13: 978-0-805-81832-1 (pbk)

Publisher's Note
The publisher has gone to great lengths to ensure the quality of this reprint but points out that some imperfections in the original may be apparent.

Contents

This page intentionally left blank

Preface

Computer support for design has gone through many generations and philosophical perspectives. In the early days, it was thought that computer-based design support should take the form of a sophisticated calculator that could handle complex analytical problems and make fewer errors than people. Similarly, there was a strong push toward CAD and computer support for generating production drawings. As the CAD systems improved and included more and more functionality, the analysis programs became capable of solving more and more complex problems with better accuracy. However, the idea of computer-based design support still eluded us.

In the late 1970s and early 1980s, there was a new approach introduced for computer-based design support: the artificial intelligence (AI) approach. Both logic and rule-based systems looked promising because they could address the aspect of design reasoning that was hard to do with numbers. Several expert systems were developed across a broad range of applications, some of which touched on the area of design. Still, these expert systems for design seemed to fall short of their promises. They needed more rules to cover more situations, they didn't learn as people did, and they were good at inferring new facts from known facts but not at synthesizing new designs. There are many reasons why the expert system approach has had limited success in providing computer-based support for design, but the point is that the research community is still looking for that elusive "computer-based design support." Professional designers are only interested if the system gets it right and saves them money.

One of the recent developments in problem solving paradigms in AI is the idea of case-based reasoning. In case-based reasoning, some aspect of a new problem provides a reminder of a previous experience and that experience can be the basis for a new solution. This approach shows great potential for computer-based design support. It combines the best of AI and expert systems, using symbolic reasoning and experience, and does not appear to have some of the pitfalls, such as the use of deduction to generate solutions, the difficulty in updating to reflect new experience, and so on. However, the use of case-based reasoning in design is in its early days, and there is some confusion about what role this new paradigm can play and how such systems are implemented.

Let's look at what case-based reasoning in design means. Assume you are a structural engineer on a design project for a new hotel that has water views on both sides of its main axis. You may recall a similar project that your firm

worked on a few years ago because it was also a hotel with a narrow floor plan. In order to reuse some of the ideas in the previous design you need to find the drawings or the reports filed after the building was completed. Or, you may contact the design engineer for the project, if he is still employed by the firm. Ultimately, in order to benefit from the previous experience of yours or of the firm that employs you, first, someone needs to remember a relevant previous project and, second, someone needs to be able to locate the information related to the relevant project.

This use of case-based reasoning, that is, to support designers when generating a new design solution, is concerned with the recall and the reuse of relevant design experience. In general, case-based reasoning can take on many faces to provide different answers to various problem solving needs. For example, case-based reasoning has been used to help generate explanations or to present experience in such a way that it teaches a lesson. In this book, the focus is on the use of case-based reasoning to support a designer in recalling a relevant design and in the reuse of that design to help generate a solution to a new design problem. In this sense the focus is oriented toward the design process view of case-based reasoning. This is a fairly strong bias and must be recognized as such. Case-based reasoning can be much more broadly defined and has been applied to demonstrate other perspectives on this problem solving paradigm.

This book has been written for academics and professionals who would like to use case-based reasoning to support designers and are looking for a specification of the process of case-based reasoning in design and alternatives for representation and implementation. The book should appeal to researchers who are considering:

- The unresolved issues of design research.
- Applications of case-based reasoning to complex problems.
- Representations of design knowledge and design experience.
- Process models of design.

From the professional perspective, this book should appeal to people involved in the development and/or use of computers during the design process. Such professionals will find the book provides a set of alternatives for the representation of design cases and examples of their implementation, a broad coverage of the issues in implementing a case-based reasoning system for design, and the advantages and disadvantages of different implementation decisions.

The book is organized into two major parts. The first part includes Chapters 1 through 5 and gives an introduction to the issues and alternatives in using case-based reasoning in design. Chapter 1 is an overview of case-based reasoning. This chapter introduces the authors' bias on the definition of case-based reasoning and introduces the terminology that will help the reader understand the rest of the book. Chapter 2 focuses on design, design processes, and the applications of case-based reasoning to support the recall and reuse of previous designs. Chapter 3 presents the considerations and alternatives for representing design cases.

Chapter 4 deals with recalling previous design cases, Chapter 5 with adapting design cases. These chapters provide a general discussion of case-based reasoning in design with examples along the way.

The second part of the book is about specific implementations of case-based reasoning in design. Rather than survey all such implementations to date, we have decided to present in detail two applications we have implemented. We present all the things we found hard to determine given the current literature on case-based design. We found many descriptions of how to represent design case memory, but found very little on how to translate this to a computer program. Hopefully, in these two chapters there are enough examples of implementations that the reader could confidently implement his or her own system. In a way, these two chapters are cases in themselves that could provide the basis for a new case-based design system. Chapter 6 presents CASECAD, focusing on the representation of design cases using a variety of representation paradigms and the strategies for recalling a similar case. Appendix A includes the actual Lisp code for CASECAD and Appendix B presents an entire design case as represented in CASECAD. Chapter 7 presents CADSYN, focusing on the contents of case memory to support design case adaptation and the implementation of a constraint satisfaction approach to adapting design cases.

In summary, the purpose of the book is to present the issues in taking a paradigm from AI, specifically case-based reasoning, showing how it is relevant in supporting designers in generating new designs, and presenting some of the implementation issues and their resolution. The book does not present an exhaustive account of how case-based reasoning can be used in design, that would take too long and be too hard. The book, therefore, focuses on something that can be done well, showing how case-based reasoning can help a designer recall a relevant previous design and reuse the design in a new design situation.

Acknowledgments. This book was written after several years of successful collaboration among the authors. The collaboration occurred at the Key Centre of Design Computing, University of Sydney. Firstly, we would like to thank the many people, including researchers, students, and staff at the Key Centre that provided useful comments and assistance in the development of the case-based reasoning projects. We acknowledge the support from the Australian Research Council and the Australian Postgraduate Research Awards Program. We thank Rita Villamayor for her assistance in collecting case data, and especially, the engineers at Acer Wargon Chapman for giving us their time and information. Special thanks to Alex Wargon for introducing us to his colleagues and persuading them to participate in this project. We thank our reviewers, including Ashok Goel and Gerhard Schmitt, for providing comments that improved the quality of the book. Finally, we thank Fay Sudweeks for producing the final copy of the book who, through her skill and patience, makes all our efforts look better.

This page intentionally left blank

About the Authors

Mary Lou Maher is Co-Director of the Key Centre of Design Computing and Associate Professor in the Department of Architectural and Design Science at the University of Sydney. She joined the University of Sydney in 1990 after 5 years on the Faculty of the Department of Civil Engineering at Carnegie Mellon University and working with the Engineering Design Research Centre. She teaches and does research in computer-aided design and knowledge-based systems. She is a Fellow of the Australian Institution of Engineers, and a member of the American Society of Civil Engineers' Expert Systems and Artificial Intelligence Committee, the American Association of Artificial Intelligence, and the IEEE Computer Society. She is the editor or co-author of six books on knowledge-based systems and over eighty papers.

M. Bala Balachandran is a Lecturer in computer science at the University of Wollongong in Australia. He received a B.S. in engineering from the University of Sri Lanka and a Graduate Diploma and a Ph.D. in design computing from the University of Sydney in 1984 and 1989, respectively. Previously, he spent five years as a research associate in the Key Centre of Design Computing at the University of Sydney. His research interests include hybrid expert system, case-based reasoning, fuzzy logic, and AI-based design. He has authored or co-authored two books and more than twenty papers on knowledge-based systems and computer-aided design. He is a member of the Institution of Engineers, Australia, the Australian Computer Society, and the IEEE Computer Society. He currently teaches primarily in the area of artificial intelligence and expert systems.

Dong Mei Zhang is a Knowledge Engineer in the CSIRO Division of Information Technology, Sydney Laboratory. She obtained B.S. and M.S. degrees in Computer Science from the Northeast University of Technology in China in 1985 and 1988, respectively. She worked as a software engineer in the China Academy of Building Research before commencing her Ph.D. at the University of Sydney. She completed a Ph.D. at the Key Centre of Design Computing in 1994 on a hybrid approach to case-based reasoning in design. Her interests include artificial intelligence in design, database development for design, and case-based design systems.

This page intentionally left blank

1 Overview of Case-Based Reasoning

Case-based reasoning promises to provide a way to support design by reminding designers of previous experiences that can help with new situations. As designers, we learn to design by experiencing design situations. For example, generating a design for a bridge requires not only an understanding of the analysis of bridges, but exposure to examples of several bridge designs. We learn to analyze through the use of formal methods, but creating a new design requires previous experience, or at least, exposure to another's design experiences. Case-based reasoning addresses this type of reminding and reuse of experience with computational models and guidelines for their implementation. In order to appreciate the role case-based reasoning can play in design, this chapter provides an overview of case-based reasoning as a computational model. In this chapter, case-based reasoning is presented simply, maybe even simplistically, in order to introduce the terminology used in the remainder of the book. For a more detailed presentation and discussion of case-based reasoning, the reader is referred to Kolodner (1993).

Case-based reasoning is an approach to problem solving that falls under the more general category of reasoning by analogy. Analogical reasoning is based on the idea that problems or experiences outside the one we are currently dealing with may provide some insight or assistance. Through analogy, we may be reminded of a window design when designing a door to a balcony with a view. Analogy is a way of recognizing something that has not been encountered before by associating it with something that has. We often use analogy to explain a concept or our reason for making decisions. Analogy can help us to search for an answer to a problem, or to explain how or why we make certain decisions, or to provide examples for teaching concepts. Even when we are not "problem solving", analogy plays an important role in understanding the world around us.

Because people are familiar with the use of analogy, it is an idea that can be studied, applied, and extended for the development of computer support for human problem solving. Analogy can be used in common situations, where the previous experience is directly applicable, or in unique or creative situations, where the previous situation shares something with the new situation, but the differences are just as interesting as the similarities. By studying the use of analogy, we can develop formal models of problem solving.

The development of formal models of analogical reasoning has been studied by researchers in AI. An early model of analogical reasoning for problem solving is reported in Carbonell (1981). The development of this model has led to new

approaches to machine learning (Carbonell, 1983), and to the distinction between derivational analogy and transformational analogy (Carbonell, 1983, 1986). Derivational analogy learns from the problem solving *process* performed in the previous experience; transformational analogy learns from the *solution* for the previous experience. The research in analogical reasoning as a means of learning new concepts from past experience has focused on the learning process and the representation of the analogy operators (Prieditis, 1988; Russell, 1989; Keane, 1988).

The application of these ideas to engineering design domains has led to the concept of structure-mapping (Falkenhainer, Forbus, and Gentner, 1990), and to the application of analogy and mutation for creative design (Zhao, 1991), providing operational definitions of analogical reasoning. Now we can consider whether the analogy draws on previous experience in the same domain or previous experience from another domain, referred to as *within-domain* or *cross-domain* analogy. The representation of previous experience determines how useful it can be in a new situation. Many of these ideas have been studied as research projects to further our understanding of how analogy can be modeled and how to operationalize the various approaches to analogical reasoning.

Seeing analogy from the perspective of memory and reminding has developed somewhat independently of analogical reasoning and has led to the concept of memory organization (Schank, 1982) as a guideline for computer representations. Continuing with the study of memory and its use as a basis for new problem solving, generating explanations, and identifying the reasons for failure has led to models for representing experience in computers (Kolodner and Riesbeck, 1986). This work has been extended, applied, and developed into an entire area of AI-based problem solving called *case-based reasoning* (Riesbeck and Schank, 1989; Kolodner, 1993). Case-based reasoning has continued as a research area within AI but has also been applied and used in real-world situations.

Case-based reasoning is a formalization for the development of a computational model of problem solving that is based on memory organization and reminding. Case-based reasoning has been developed as a process model with specific stages and knowledge resources that reflects the research in analogical reasoning. The relevance to designers is to apply the research in memory organization for defining a case memory of previous designs and to apply the process of analogical reasoning for the reuse of previous design experiences.

This chapter presents a view of case-based reasoning as an operational model of problem solving that supports a designer in generating a new design concept. Two extreme views of case-based design are:

1. To provide a resource of previous experience to aid a designer, called a *design aiding system.*
2. To provide a computational approach to the design process, called a *design automation system.*

Because most applications of case-based reasoning tend to fall somewhere between the two extremes, the focus of the presentation in this book is *design support*. In order to understand how case-based reasoning can support design, we first overview case-based reasoning and its application more generally. Then from this common base, we can build the concepts and issues related to case-based design in subsequent chapters.

CASE-BASED REASONING AS A COMPUTATIONAL PROCESS

Case-based reasoning is an approach to problem solving that makes use of a database, or case base, of previously solved problems when solving a new problem, where a database is a collection of data stored in the computer, and a case base implies that the data represent previous problem solving episodes. This approach is used as a model to guide the development of computer programs that assist in problem solving by accessing the case base directly. The concerns of case-based reasoning then become recalling the relevant case or cases and reusing or adapting the relevant cases.

Case-based reasoning as a computational model of problem solving is contrasted with the expert systems approach. Both approaches rely on the explicit symbolic representation of knowledge based on experience to solve a new problem. Expert systems (Jackson, 1990; Buchanan and Shortliffe, 1984) use past experience stored in a knowledge base of generalized heuristics to assist in solving a new problem. The generalized heuristics can be stored as rules of thumb or as logical inferences. Case-based reasoning uses a representation of specific episodes of problem solving to learn to solve a new problem. Both case-based reasoning and expert systems use the experience of past problem solving when solving a new problem. Case-based reasoning systems store past experience as individual problem solving episodes, and expert systems store past experience as generalized rules and/or objects.

For example, the development of a computer program that assists in the design of a new pedestrian bridge over a busy street could be based on an expert systems approach or a case-based reasoning approach. An expert systems approach would encode heuristics about the types of bridges and their appropriateness for the span and width of the crossing, among other relevant information about bridges. A case-based reasoning approach would have a case base of previously designed bridges, among which one or more would be selected as the starting point for the new bridge design. Using the expert system approach, a new bridge design would be generated by applying the relevant rules to define the parameters of the new bridge. Using the case-based reasoning approach, a relevant previous bridge design would be recalled and adapted to the new design situation.

In many ways, a case-based reasoning approach mimics the way a person may solve a problem by recalling a previously solved similar problem. The person is reminded of the previously solved problem because it has some relevance to the new problem. In the bridge design example, the person may

recall another pedestrian bridge, or maybe a bridge over the same span. After the person recalls a previously solved problem, certain aspects of the previous problem's solution are used in the new context and others are not. In the bridge design example, a pedestrian bridge with a span of 15 meters may be recalled, but the new design requires a span of 18 meters. The same design for the superstructure, such as the steel arch, may be used, but the span and sizes of the steel members will change.

The two major considerations in case-based reasoning are:

1. Identifying a relevant previous experience.
2. Determining what changes and what stays the same.

Identifying a relevant previous experience can be considered as searching the case base for a match with some predefined features or attributes of the new problem. For example, the pedestrian bridge design problem may use a search for other pedestrian bridges of a specified length. Determining what changes and what stays the same can be considered as adapting some of the attributes of the retrieved case to fit the new problem. For example, adapting a pedestrian bridge in case memory with a length of 15 meters for a new bridge with a length of 18 meters requires changing the length and other related attributes.

Modeling these two parts of case-based reasoning as a computational model is illustrated in Figure 1.1. The two major considerations are referred to as *recall* and *adapt*, where during recall the case base is searched for a relevant previous experience, and during adapt the decisions of which aspects to change and which to retain are made. Using this model for problem solving, the system can learn from its own experience, as well as the experience of the people who supply the cases. Each new solution can be added to the case base, making it available in a new problem solving session.

Looking more closely at the model of case-based reasoning illustrated in Figure 1.1, the two processes, recall and adapt, can be defined in more detail. The recall process can be decomposed into indexing, retrieval, and selection. Indexing is concerned with the identification of the attributes that a previous problem should have in order to be relevant to the new problem. Retrieval is a process of identifying which cases that have all or a subset of those attributes should be retrieved for further consideration. Finally, selection is a process of evaluating the retrieved cases so they may be ranked. The "best" case is then selected for adaptation.

The adapt process includes a recognition of the differences between the selected case and the new problem and decisions regarding what aspects of the case are changed to fit the new problem. This process can be decomposed into *modify* and *evaluate*. Modifying a selected case is a process of changing parts of the case description. Evaluating the modified case is a process of checking the new case description for feasibility. The elaborated case-based reasoning process is illustrated in Figure 1.2.

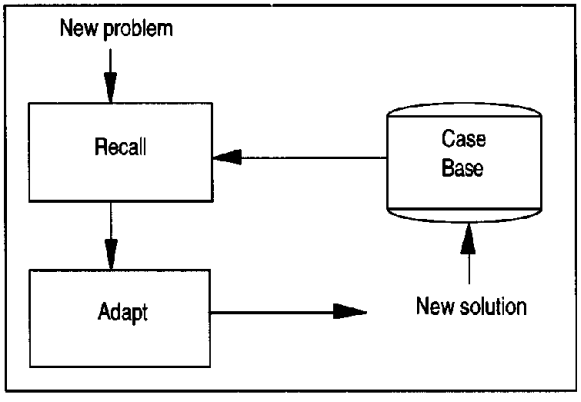

Figure 1.1. A simple model of case-based reasoning.

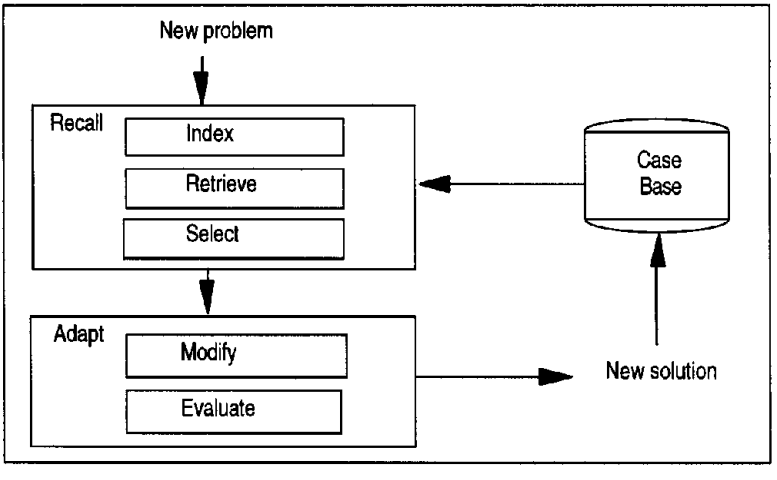

Figure 1.2. An elaborated model of case-based reasoning.

Implementing a case-based reasoning approach as a computer program follows the models already described, where each component of the model is an algorithm or database. The overall algorithm for case-based reasoning (CBR) can be summarized as:

```
begin
  get new problem specifications;
  identify indexing attributes;
    retrieve a set of cases that match attributes;
    select one case;
    repeat
      modify case;
      evaluate solution;
    until solution is satisfactory;
end
```

There are several approaches to implementing this algorithm. For example, to get new problem specifications, the program may simply ask the user to type in a set of specifications and use the specifications directly as the indexing attributes, or it may reason about the user's unstructured specification to identify key attributes for searching case memory. Another example of the different ways CBR can be implemented is to *select one case* that has the largest number of attributes with the same values as the new problem or the case that has the highest ranking in a weighted count of matching attributes. Alternatively, recall could be less structured, as the user navigates through a case base without providing the new problem specifications or selecting one case to consider further.

The use of a CBR approach requires the definition of a representation of the case base, or *case memory organization.* This representation defines how the reasoning can be implemented. The content and organization of cases determines their reuse in a new situation. A very simple example of this is in the bridge design problem: If the case base does not include the attribute for the length of the bridge, then the new problem cannot be matched against this attribute. However, for adaptation purposes, if case memory does not include knowledge about how a change in length affects the size of the bridge components, then the change in length to match the new problem cannot be evaluated.

CASE MEMORY

Case memory, or the case base, includes a representation of a set of previously solved problems. This representation provides the basis for their use by the CBR system and by the person using the CBR system. The importance of determining what is in case memory and how this is represented cannot be overemphasized. The computational process of CBR assumes an adequate and useful case base. The development of case memory is an ill-defined process that can be as difficult as the knowledge acquisition stage in developing an expert system. In fact, the case base is the equivalent of a knowledge base for CBR.

This section provides a brief overview of the issues and alternatives in developing case memory. These issues and others are considered in more detail in subsequent chapters, with special consideration to design. Developing a case memory includes a definition of:

- The content of each case.
- The representation of the contents of each case.
- The organization of the set of cases in case memory.

The content of each case is basically a description of the previously solved problem. The difficulty in translating this into a representation of case memory in the computer lies in the identification of relevant information to include in the description. The problem solving episode to be recorded in case memory may have well-defined problem specification and solution attributes or a well-defined process by which it was solved. Formalizing this aspect of CBR requires an analysis of the problems for which the CBR system will be used, as illustrated in Figure 1.3.

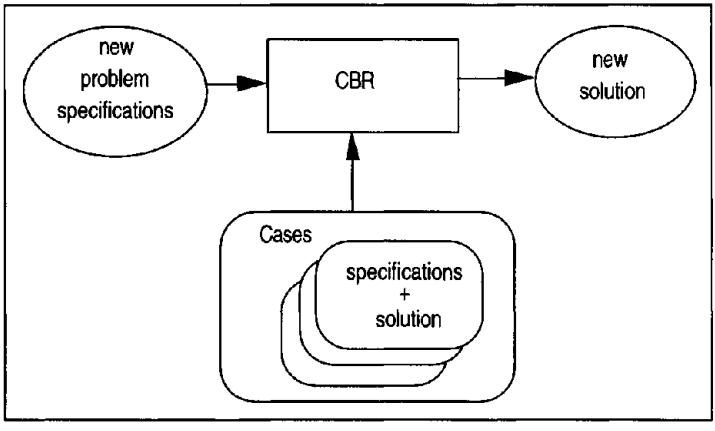

Figure 1.3. Analyzing the contents of case memory.

In order for there to be some commonality between the specifications to the new problem and the individual cases in case memory, a description of a case should include a representation of the initial specifications. In order for the cases to be useful in solving the new problem, a description of a case should include a representation of the solution. This is the most simplistic analysis needed for defining the content of case memory. In addition to the basic specifications + solution descriptions, case memory will also need to include knowledge for identifying a similar situation when the new problem specification is incomplete. This could take the form of indexing features and priorities. Case memory may also include a description of intermediate stages in the development of the solution, and possibly assumptions and justifications associated with decisions in the previous problem.

The representation of the contents of each case defines how the information about a case is organized, as a set of attribute-value pairs, as part-subpart

relationships, or as a network of attributes. These three alternatives are illustrated in Figure 1.4. Representing a case as attribute-value pairs represents the relevant decisions in the previous case and the specific values associated with each decision. Representing a case as a hierarchy of part-subpart relationships facilitates representation and reasoning about large and/or complex cases. The hierarchical representation includes more content than the attribute-value pairs representation because it includes relationship knowledge. The network-based representation of cases can build on the hierarchical representation with multiple attribute-value pairs at each node and allow additional types of relationships to be represented, or it can be similar to a semantic network, where the nodes in the network represent a single feature of the case.

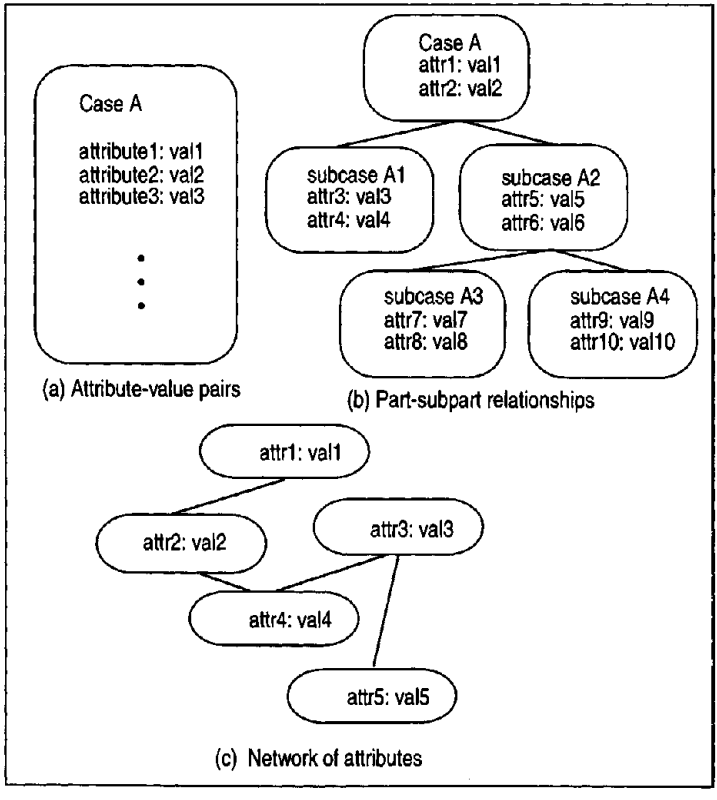

Figure 1.4. Organization of a case in case memory.

The representation of a case is usually generalized for all cases in case memory, so that all cases are described by the same set of attributes or part-subpart relationships, or all cases are described as networks of attributes. In this

way, the organization of cases in case memory provides a template or model for defining the content of a case and for adding new cases to an existing case memory. This stage in the development of a CBR system results in a clearer definition of the contents of case memory and the representation schemes relevant to a CBR solution to a class of problems.

The organization of the set of cases in case memory provides mechanisms for locating one case or a part of a case in case memory. The cases may be clustered or accessed by common attributes. Organizing cases in a structured way allows the CBR system to quickly and accurately search for similar cases. As case memory becomes very large, the need for an organizational structure becomes more important. The simplest way to organize case memory is to store the name of each case in a list with a pointer to the content of the case. As case memory gets larger, the use of an indexing tree, where each node in the tree is an attribute-value pair of one or more cases, reduces the space that needs to be searched. These two approaches to organizing case memory are illustrated in Figure 1.5. In Figure 1.5 (b), cases B and C both have attr4: val4, attr1: val1, and attr2: val2.

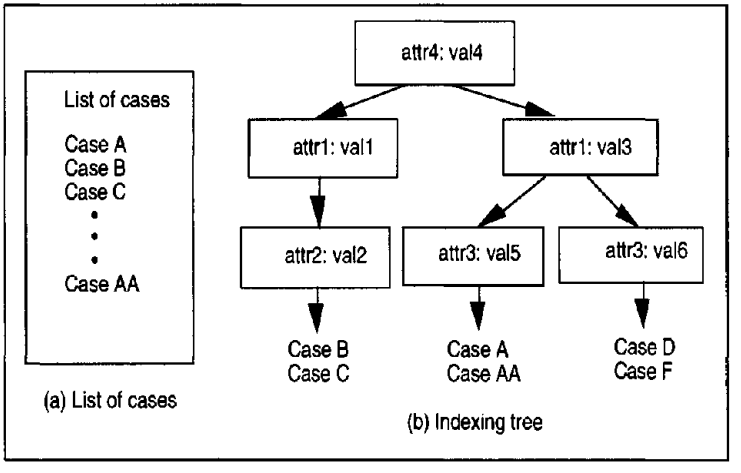

Figure 1.5. Organizing a set of cases in case memory.

The organization of case memory for access to an individual case is referred to as the *indexing scheme*. The development of an indexing scheme is dependent on the size of case memory and how a new problem is specified. If case memory is not large, with less than 50 cases, the indexing scheme that uses a list of cases is sufficient. Retrieving a case from a larger case memory would be too slow with the list of cases scheme. In Figure 1.5, the indexing scheme illustrated in part (a) assumes that all cases will be retrieved by name and that each case will be compared separately to the new problem specifications. The indexing scheme

illustrated in part (b) assumes that the attributes in the nodes of the indexing tree will be part of the specification of the new problem. If these attributes are not part of the new problem specification, then the relevant set of cases cannot be identified.

RECALLING CASES

Recalling a case from case memory is a pattern-matching problem that is based on the specification of a new problem. For example, the specification for the design of a new building is recorded in a document called a *brief.* A subset of the brief may be useful in recalling one or more similar building designs. A new problem may be specified as a set of attribute-value pairs, as a set of constraints or conditions on the values of each attribute, or as a network of attributes. Given a specification, the recall process can be decomposed into three tasks:

1. Index.
2. Retrieve.
3. Select.

In order to index cases in case memory, the specification of a new problem is transformed into a pattern to be matched. The pattern may be taken directly as the user input specification or it may be modified, for example, to include an order of importance of the attributes. A general approach to determining a pattern to match for indexing case memory for a specific new problem is given below.

```
begin
  identify features of specification;
  organize features;
    begin
      identify critical features;
      assign weights to features;
      group features;
    end
  output pattern for matching;
end
```

The retrieval task in CBR searches case memory for matches between individual cases and the pattern that serves to index the cases. Each case in case memory may be compared to the pattern, or the pattern may provide a set of indices to partition case memory, so only a relevant subset of cases are compared with the pattern. Retrieval can be based on a perfect match, where the pattern is found exactly, or on partial matches. If partial matches are retrieved, a threshold may be set to determine when a partial match is close enough. A general approach to case retrieval is given below.

```
begin
  get pattern for matching;
  determine relevant cases for comparison;
```

```
    repeat for all relevant cases
       determine how close the case matches the pattern;
       if case match above threshold
          then add case to retrieved list;
    output list of retrieved cases;
end
```

The selection task in CBR orders the retrieved cases to determine which case is the best match. The selection process depends on the pattern used to index case memory. If the pattern is a set of weighted features and each retrieved case is ranked according to the weight of the matching features, selection is based on the retrieved case with the highest ranking. If the pattern is simply a set of features, then selection is based on the case with the most features in common with the indexing pattern. Selection is the result of ranking the retrieved cases, where there are various methods for determining the ranking.

ADAPTING CASES

In some problems, a selected case provides a solution to the new problem. For example, when using a CBR approach to determine if a loan should be approved, the retrieved case provides a solution: approve or disapprove. In most problem solving, however, the selected case needs to be modified to be appropriate as a solution to the new problem. For example, a CBR approach to designing a bridge may recall a similar bridge, but the recalled bridge design cannot be used without modification for the new site. This modification is referred to as *adapting a case.*

Adapting a case from case memory to solve a new problem requires additional knowledge. The form this knowledge takes depends on how adaptation is done. One approach to adaptation is to identify those attributes of a case that can be changed and associate a formula with each adaptable attribute to be evaluated during adaptation. Another approach to adaptation is to associate a set of constraints with case memory that must be satisfied when adapting a case. These two approaches are broadly defined as *parametric adaptation* and *constraint satisfaction.*

Parametric adaptation adapts a previous case for a new problem solving episode by associating formulas with the attributes, or parameters, that can be changed. When defining the content of case memory, certain attributes are identified as adaptable attributes. Each of these attributes has a formula or procedure to be evaluated during case adaptation. The general approach to parametric adaptation is given below.

```
begin
   repeat for each adaptable attribute
      evaluate associated formula;
end
```

This approach to case adaptation assumes that the adaptable attributes are the same for each case retrieved and for each new problem encountered. The parametric adaptation considers each attribute in isolation from the others, so the change in value of the attribute is not dependent on other adaptable attributes. The consistency of the solution as a collection of attributes is not checked.

The constraint satisfaction approach to case adaptation requires a set of constraints to be associated with case memory. Once a case is selected as the basis for the new solution, the constraints provide the knowledge needed to check for inconsistencies between the selected case and the new problem. A domain of possible values for each attribute is needed when changing the values of attributes to satisfy constraints. The approach to constraint satisfaction is given below.

```
begin
  propose new solution;
  check constraints;
  repeat
    change values of attributes of proposed solution;
  until all constraints are satisfied;
end
```

Adapting cases is a potentially complex task that has not been fully explored. When possible, a simple approach to adaptation is used to automate the adaptation process. Otherwise, the user of the CBR system assists, or even performs, case adaptation.

APPLICATIONS OF CBR

CBR technology has been applied to a wide range of problems. CBR systems have been demonstrated in such varied domains as law, medicine, cooking, dispute resolution, customer service, labor mediation, process control, and engineering design, especially in the past several years, yielding significantly improved performance and cost savings. CBR provides the potential for developing knowledge-based systems more easily than generalized knowledge-based approaches. The assumption is that it is easier to produce specific examples of problem solving than to generalize across a class of problems to suit all new problems encountered.

Some case-based systems have been developed to solve problems automatically, whereas others have been built only to aid human problem solvers. CBR systems deployed in the field of industrial and commercial applications have delivered a number of benefits, including reduced cost, reduced errors, faster system development, reduced training time, decreased personnel required, improved decisions, and improved customer service. Although a wide range of potential applications have been explored, commercial CBR applications have focused, for the most part, on case retrieval for decision support.