Capturing Design Knowledge

Brian A. Babin and Rasiah Loganantharaj

Center for Advanced Computer Studies
University of Southwestern Louisiana
P.O. Box 44330, Lafayette, Louisiana 70504

Abstract

This paper proposes a scheme to capture the design knowledge of a complex object including functional, structural, performance, and other constraints. Further, the proposed scheme is also capable of capturing the rationale behind the design of an object as a part of the overall design of the object. With this information, the design of an object can be treated as a case and stored with other designs in a case base. A person can then perform case-based reasoning by examining these designs. Methods of modifying object designs are also discussed. Finally, an overview of an approach to fault diagnosis using case-based reasoning is given.

1 Introduction

At the abstract level a design task involves finding a consistent assignment for a set of variables that together define the object desired and satisfy the functional, structural, performance, and other constraints placed upon the object. In other words, a design task involves solving a constraint satisfaction problem where the constraints define the functional, structural, performance, and other requirements of the object[6]. Unfortunately the constraints defining a new object are often incomplete, ill-defined, or inconsistent. In such situations the design process involves the iteration of the following steps: refinement of the object specification followed by partial constraint satisfaction[2][3]. This process is repeated until a complete specification of the object is achieved.

The design process can be systematic, as we briefly discussed, or ad hoc. Unfortunately many of the designs to date were developed in an ad hoc manner. The method of systematic design is not very well understood. As a result, there have been a number of recent efforts towards a better understanding of the design process[3][6][10][11].

In this paper we are interested in capturing the design knowledge of an object. The purposes are twofold: (1) capturing the design knowledge makes it much easier

to understand, modify, or enhance the design, and (2) studying the designs of objects and the knowledge which needs to be represented will help in systematizing the design process.

In order to understand the design knowledge and representation issues involved, we have chosen to study the process of designing complex objects. Loosely stated, a complex object is anything with a nontrivial design consisting of a number of parts and their interconnections. Examples of complex objects include toasters and table lamps at the relatively simple end and jet engines at the complex end. Hereafter a complex object is simply referred to as an object.

When representing an object we need to capture the decompositional, hierarchical, functional, structural, and physical knowledge for that object[1]. We also need to capture the design knowledge including the decisions made while designing the object, the rationales behind these decisions, and any alternatives that were considered. Given the design of an object, it may be relatively easy to capture the decompositional, hierarchical, functional, structural, and physical knowledge of that object. However, the acquisition of design decisions, alternatives, and rationales is likely to be quite challenging, especially for an ad hoc design.

All of this information about an object can be gathered together to form a particular *design experience.* A designer may study this design experience while designing another object which is similar in some respect. The desire to examine past design experiences makes case-based reasoning (CBR) a natural way to access these experiences. CBR can be applied here by treating a design experience as a case and building a case base of various design experiences. Background information on CBR or, more generally, memory-based reasoning can be found in [5], [7], and [9]. Designs can be indexed according to parts used, functionality, and other features. This allows a designer to examine the design experiences of other objects with similar parts or functionality to help in making design decisions. We also briefly address the issue of diagnosing faults which cause an object to malfunction. We employ CBR to find the cause of a malfunction by examining the causes of previous similar malfunctions.

The rest of the paper is arranged as follows. First, the physical representation of an object is described since it is the object itself upon which everything else is based. Next, the method of designing an object from constraints is discussed. The design process produces not only the physical representation of an object, but also *why* it was designed the way it was and *how* it satisfies its constraints. Then various types of design modifications which may be required or desired are examined. Finally, a brief look is given at fault diagnosis for complex objects.

2 Object Representation

Central to the idea of representing the design of a complex object is the representation of the object. Indeed every other aspect of the design is at least indirectly dependent

on the representation of the object. For example, the proof that an object has certain desirable properties will likely contain a reference to the properties of a physical component in the design.

There are two alternatives for representing an object: (1) have separate conceptual and physical decompositions of an object with a mapping between them, or (2) combine the conceptual and physical decompositions into a unified representation. When the conceptual decomposition of an object matches with its physical decomposition, the cognitive complexity of the system is reduced and hence the representation of the object becomes easier to understand. Therefore we favor matching the conceptual and physical decompositions of an object. To capture the relevant design knowledge, we should be able to represent the decompositional, hierarchical, functional, structural, and physical knowledge for the object[1]. We use a frame-based scheme to capture the above design knowledge since it is capable of representing the required information and provides an easy means for manipulating this information.

Decompositional knowledge of an object is represented in a hierarchy. An object is composed of parts, each of which can be composed of subparts, and so on. A collection of subparts may be thought of as a component of the object. Relationships between parts in the hierarchy are represented by using HAS-PART and PART-OF links. An object is ultimately realized by a collection of parts which are elementary or are themselves complex objects with their own designs. A part is considered elementary if it is essentially nondecomposable. Examples of elementary objects are nails, screws, sheet metal, and glass. Physical knowledge is represented by describing elementary objects in terms of size, shape, color, composition, mass, etc. This scheme allows different objects to use the same part in their designs without having to each give the physical representation of that part. It also allows the physical design of an object to be examined in as much detail as is desired.

To take a simplified example, a lamp can be decomposed into the following parts: base, cord, switch, shade holder, and shade. The switch is an example of a complex object which has its own design. The base can be partially described as being white, ceramic, cylindrical, and 12 inches tall.

The structural knowledge of an object is represented by explicitly specifying the interconnections between the parts. This provides among other things a method of specifying the orientation of one part with respect to another and the type of connection (i.e. glue, weld, interlocking parts) between parts. The interconnections between parts are represented by links. The specification of an interconnection link is necessarily complex because of the nature of the relationship being represented. Not only must the type of connection between parts be specified, but the spatial orientation between connected parts must be specified as well. To use the lamp example, it can be specified that the bottom of the switch screws into the top of the base. Thus the physical design of an object is represented by a hierarchy of parts and their interconnections.

Functional knowledge is represented by describing an object in terms of what it

does. As mentioned earlier, every object serves some purpose. For example, the purpose of a toaster is to toast things. The exact representation required to represent functional knowledge has not yet been decided upon.

The taxonomy of objects is represented by classes and subclasses using IS-A and SUBCLASS links, respectively. A particular object is an instance of some class of objects. Thus objects which are similar can be located and compared. The collection of the object designs forms the case base. An indexing scheme is needed to be able to locate and examine objects with specific characteristics. This may be useful when designing an object since choices of already existing objects can be examined by the designer when deciding what to use for a particular part. Also, the PART-OF links allow the designer to analyze how a part was used in other designs. Using this information, the designer can decide whether an existing object will suffice or a new object must be designed. As is described later, a new object may be created by modifying the design of an existing object.

3 Designing from Constraints

The design of an object provides a consistent assignment for a set of variables that together define the desired object and satisfy the constraints placed upon that object. General classes of constraints include functionality, cost, performance, size, and weight. An example of a cost constraint is "The lamp must cost no more than 30 dollars at current prices." Obviously there are many other classes into which constraints may fall.

The constraints for an object are specified first and then a design for the object is developed to meet these constraints. There exists a gap between the constraints and the design since there is often no reference to how each constraint is satisfied in the design. Some sort of bridge is needed to link the design of an object with the set of constraints which it must satisfy. This is especially important when a new object with slightly different constraints from an existing object is desired. By identifying the parts of the existing design dependent on the altered constraints, the new design can be obtained by modifying the existing design to meet the new constraints.

As the complexity of an object grows, so do the number of constraints. A structured form of specifying constraints is needed to manage complex descriptions. The method presented here uses top-down constraint refinement and bottom-up constraint satisfaction, similar to the plausibility-driven design method described in [3]. Indeed, the latter method will be used as a basis in formally developing the former method.

Initially, very general constraints are given for an object. These constraints, referred to as *top-level* constraints, generally include properties of the object as a whole. Each top-level constraint is then refined into more specific constraints. It must be shown that satisfying all of these more specific constraints will cause the top-level constraints to be satisfied. Thus if constraint A is refined into constraints X, Y, and

Z, it must be shown that A is satisfied if X, Y, and Z are all satisfied. Each of the more specific constraints are then refined into even more specific constraints in the same fashion. The refinement process continues until sufficiently precise constraints are reached. These constraints are referred to as *elementary* constraints. An elementary constraint corresponds to a feature which must be present in the design of the object. An example of an elementary constraint is "The lamp switch must have the capacity for a 100 watt light bulb". An elementary constraint is satisfied if the design contains that particular feature. Because of the nature of the constraint refinement, the top-level constraints are satisfied if all of the elementary constraints are satisfied.

Once the constraints for an object have been specified, the design of the object is then developed to meet these constraints. The designer specifies the design of the object as a hierarchical decomposition of parts and their interconnections. In order to prove that the object designed satisfies its constraints, it is necessary to show how each elementary constraint is satisfied in the design. These relationships are also needed to indicate the constraints with which a particular component of the design is involved. This allows modifications to be made to the design without violating constraints which were already satisfied.

Many of the major decisions made while designing an object regard determining how the object should be decomposed to form its design. Some decisions concerning the structure of the object may be made when the constraints for the object are specified, however most decomposition decisions will probably be left to be made when the design of the object is created. The physical decomposition of a similar previously defined object may be used as a guide to decompose the object currently being designed. The designer may instead choose to use selected components of the physical decompositions of other objects in places in the current design. It is also possible for the designer to incorporate innovative ideas into the current design. Thus the designer has two sources of knowledge regarding how an object can be decomposed – previous designs and innovative thinking.

It is therefore important for a designer to be able to access and examine the designs of existing objects to help in making design decisions. However, this may still leave a critical gap in the information needed by a designer. This gap is caused by the fact that while a design describes the physical structure of an object, the rationale behind the design is often not represented. Capturing the knowledge involved in designing an object is of great importance in many areas, especially where the lifetime of the project is expected to exceed the time of involvement of the designers, as in [4]. Thus it is crucial to be able to retain not only the design of the object but also the reasoning behind the design so that it can be referenced in the future. This problem can be solved by including the decisions made while designing an object with the actual design of the object. By having the rationale behind previous designs as well as the designs themselves, a designer can make more educated decisions regarding the object being designed.

When the design of an object is being created, there are usually many ways in which various constraints can be satisfied. Thus at any given point in the design the

designer may have a number of alternatives in choosing which part to use or how a particular component should be decomposed. When a decision is reached, it is important for the designer to document the object design with the justification for that decision. This justification should include the choices considered, the reason why the decision was made as it was, and reasons why other choices were not selected.

For example, while designing a lamp, a designer needs to decide on the switch to use given the constraints "The lamp has a two position on-off switch" and "The lamp switch must have the capacity for a 100 watt light bulb". By examining various types of light switches available, the designer narrows the choice down to three switches – a cheap two position sliding bar switch, a slightly more expensive two position sliding bar switch, and a two position rotating switch. The designer is then informed that for a two position switch, the sliding bar type is preferred over the rotating type of switch. The third switch is ruled out because of this fact. The designer then chooses the first switch citing the fact that it is less expensive than the second switch. However after tests show that the second switch is considerably more durable than the first switch, the designer chooses to use the second switch.

The justification of the decision to choose the second switch is stated as follows: (1) In study X, it was found that the sliding bar type of switch is the preferred type for two position switches. The rotating switch is ruled out because of this fact. (2) The cheap sliding bar switch is chosen initially because of its cost. (3) Test Y showed that the slightly more expensive sliding bar switch considerably outlasted the cheap sliding bar switch. The former switch was then chosen replacing the latter switch because of its quality.

In this way the evolution of an object including the rationale behind the design can be captured in an integrated form which can then be analyzed and/or critiqued by others. With the justifications, the design can be more easily understood by persons not involved in designing the object.

It is important to note that objects which are not designed in the manner outlined here can still have their designs represented. The design would only consist of the physical design of the object as described in the previous section along with whatever constraints and design decisions are available. This information could be obtained from the available design documentation and supplemented by interviewing the designers involved in the project. Without this, all previously designed objects could not be represented, thereby rendering this entire scheme useless.

4 Design Modification

Some time after the design of an object is completed and judged to satisfy all of its constraints, there may be a need to make modifications to the design. Modifications may be necessary because a problem arose with the original design, some constraints were left out of the original design, an improvement can be made, or an enhancement

to the original design is desired.

Through the constraint satisfaction method every effort is made to insure that the design developed satisfies the constraints imposed on the object. In developing the design of an object, the designer will likely use parts which have been previously designed and whose functionality matches what is needed in the design being developed. If it is discovered that the functionality of a part does not meet what was claimed, a problem may arise in objects which use the part. If such a problem does occur then the design of objects which use the part must be modified in order to satisfy the constraints which were violated as a result of the part not performing as expected.

It may be the case that some constraints were left out of the original design. The constraints may have been either overlooked or not thought to be important. As a result, the design of the object turned out to not quite match what was desired. Thus the constraints omitted are added and the object is redesigned to incorporate the new constraints.

The design of an object can be modified to reflect an improvement is some area(s) of the design. For example, a new part may become available which can replace a part in the original design and is cheaper, faster, smaller, or more reliable than the part originally used. Thus if an inexpensive switch which never wears out is developed, it can be substituted for the switch currently used in the lamp design. An improvement to the design can be accomplished by changing the constraints referencing the old part to reference the new, improved part. The new part must still satisfy those constraints. The reference to the old part is retained in order to reflect the history of the design.

Modifications can also be made to enhance the functionality of a design. For example, the simple on-off switch on the lamp may be replaced by a three-way switch to produce a more versatile lamp. Constraints indicating the enhancement are added where appropriate and the design of the object is altered to satisfy the new constraints. It must be insured that previously satisfied constraints remain satisfied after the design modification is made.

The difference between these last two types of modifications is that an improvement makes a design better without changing its basic functionality while an enhancement extends the functionality of an object (but does not necessarily make it better).

As is the case in the process of creating the original design, justifications of design modifications should be included in the design. This is necessary to maintain the complete history of an object design.

Modifying a design need not always replace an old design with a new one. Instead, the old design must be allowed to exist as an object as long as it is still useful. This is especially the case regarding enhancements to a design. The old and new designs may remain interconnected if desired so that a change in one effects a change in the other. On the other hand, the old and new designs can be made totally independent of one another if significant changes are made to the old design.

5 Fault Diagnosis

A common characteristic of every complex object is functionality – each complex object serves some purpose. For example, the function of a toaster is to toast things and the purpose of a lamp is to provide light. The design of an object is such that the functionality desired is achieved. However it may be the case after some period of time that the object does not function properly, i.e. the toaster does not toast or the lamp does not provide light. In other words, there is a fault associated with the object. The fault must be diagnosed in order to correct the problem. A fault can be diagnosed using either deep level reasoning[8] or shallow level reasoning.

One method of diagnosing a problem is by using deep level reasoning to analyze the design of the object and determine the cause of the malfunction. Thus by knowing that the lamp does not provide light, analyzing the design of the lamp will discover that the fault is caused by a burnt bulb, a broken switch, or a bad cord. Note that the function of the light bulb must be included in the design of the lamp since it is the bulb which actually produces the light. This method by itself may have difficulty dealing with nontrivial malfunctions where the failure of one part leads to failure of others. In any case, it is desirable to avoid having to analyze the entire structure of an object every time a malfunction occurs.

Another approach to problem diagnosis is to use case-based reasoning to find similar malfunctions which have occurred previously. This approach uses shallow level reasoning since it does not try to reason from the physical design of an object, only from past experiences with malfunctions. If a similar malfunction has already occurred with that object, the corrections used to eliminate the previous malfunctions are examined. If a successful correction is found and the circumstances are similar enough, the correction is tried on the current problem. If the circumstances are not quite the same, the correction may have to be adapted to apply to the current problem. The success or failure of the correction is stored along with the circumstances under which the correction was applied. In the case of a failure, an explanation of why the correction failed (if known) is also stored.

If no cases of malfunctions with similar circumstances are found for the object, similar objects can be examined. If possible, analogical reasoning could then be used to adapt the corrections applied to one object to apply to the current object. If no past experience can be used in the current situation, a diagnosis by an expert or one based on the design of the object (using deep level reasoning) must be formulated. As time goes by and varied types of malfunctions occur, the case base grows and the fault diagnosis capability for an object improves.

6 Conclusion

This paper introduces a design scheme which integrates the process of designing complex objects within a framework that allows for the capture of the design knowledge

that went into the design. The framework is intended to be sufficiently general so that any object can be represented. The presence of a broad domain model eliminates most of the redundancy and wasted effort caused by the inability to integrate rigidly defined domains.

Case-based reasoning is used to provide designers with knowledge of parts, past designs, and the rationale behind these designs to assist in the design process. CBR is also used to help diagnose problems which occur in an object while it is in use.

Future work will be focused mainly on formalizing many of the ideas presented in this paper. The process of designing objects will be investigated further to provide more insight into what is required to fully capture the knowledge utilized when designing an object.

References

[1] Addanki, Sanjaya and Ernest Davis. A representation for complex physical domains. In *Proceedings of the Ninth IJCAI*, 1985.

[2] Aguero, Ulises. *A theory of plausibility for computer architecture designs.* Ph.D. dissertation, Center for Advanced Computer Studies, Univ. of Southwestern Louisiana, 1987.

[3] Aguero, Ulises and Subrata Dasgupta. A plausibility-driven approach to computer architecture design. *Communications of the ACM*, 30(11):922–932, November 1987.

[4] Freeman, Michael S. The elements of design knowledge capture. In *Proceedings of the Fourth Conference on Artificial Intelligence for Space Applications*, November 1988.

[5] Kolodner, Janet L. Extending problem solving capabilities through case-based inference. In *Proceedings of the 4th Annual International Machine Learning Workshop*, 1987.

[6] Mostow, Jack. Toward better models of the design process. *AI Magazine*, 6(1):44–57, 1985.

[7] Schank, Roger C. *Dynamic Memory – A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, Cambridge, England, 1982, Chapter 2.

[8] Sembugamoorthy, V. and B. Chandrasekaran. Functional representation of devices and compilation of diagnostic problem-solving systems. In Janet L. Kolodner and Christopher K. Riesbeck, editors, *Experience, Memory, and Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, Chapter 4.

[9] Stanfill, Craig and David Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, December 1986.

[10] Steele, Robin L. Cell-based VLSI design advice using default reasoning. In *Proceedings of the Rocky Mountain Conference on AI*, 1988.

[11] Steinberg, Louis I. Design as refinement plus constraint propagation: the VEXED experience. In *Proceedings of the Sixth AAAI*, 1987.