

What's in a case: the use of case bases, knowledge bases and databases in design

M. A. Rosenman, J. S. Gero, R. E. Oxman[†]

Design Computing Unit
Department of Architectural and Design Science
University of Sydney NSW 2006 Australia

[†]Department of Architecture
Technion Israel Institute of Technology
Haifa 32000 Israel

Design experience can be classified into generalised or compiled knowledge and specific knowledge. Generalised design knowledge has been introduced into computer-aided design in the form of rules, frames and more recently, design prototypes. Case-based reasoning is a well-defined paradigm in artificial intelligence and has obvious scope for its use in design reasoning. This paper explores case-based reasoning in design and argues for the integration of both specific and generalised design knowledge. This integration allows for characterising what is in a case by drawing upon the schema developed for design prototypes. Finally, the paper argues that the addition of precedent knowledge, in the form of case bases, to knowledge bases and CAD databases will further extend the experience-based capabilities of design systems.

Introduction

Computer-Aided Design

Computer-aided design has been identified with the uses of databases. These databases contain syntactical information regarding the topological and geometrical properties of the artefacts under consideration. Occasionally, they contain other syntactical information such as material properties. They may allow for the description of classes of objects as well as instances. All semantic information is implicit and is obtained by users' interpretation of representations of the database using their own interpretive knowledge. Recently, knowledge bases have been introduced into computer-aided design as a means of representing generalised or compiled design knowledge (Balachandran et al, 1990). These knowledge bases augment rather than displace databases by providing the semantic information missing from CAD databases. This paper intends to show that case bases add specific precedent knowledge and, by augmenting rather than displacing knowledge bases, further extend the capabilities of such integrated systems.

Modeling Design Experience

Experience plays an important role in design reasoning. Experienced designers use previous designs to solve current situations, as in the use of standard details. One of the great challenges of knowledge-based design has been to develop models of reasoning processes based upon experiential knowledge in order to support routine and non-routine design. Current approaches to design reasoning are based upon the representation of compiled knowledge employing rules, frames and schemas such as design prototypes

(Gero, 1987; Rosenman and Gero, 1989). However, the knowledge of specific design histories may have significance in adaptive design.

Compiled knowledge is that knowledge in which several like experiences are generalised into rules or rule-like procedures wherein the details of the specific experiences are usually lost. This compiled or generalised knowledge covers a range of situations rather than a specific situation and its application is contingent on focussing onto a specific situation.

Case or specific knowledge is that knowledge in which specific experiences or episodes are kept and reasoning is based on retrieving the most appropriate experience rather than repeating the process to create it. Case-based reasoning from precedents is well understood in the field of law and is a well-defined paradigm in artificial intelligence (Riesbeck and Schank, 1989; Kolodner, 1987). There are obvious parallels to the use of precedents in design.

The cognitive significance of the relationship between generalized knowledge, such as the design prototype and the detailed knowledge of specific experiences is a current theme in artificial intelligence research (Alterman, 1986).

Towards an Integrated System

This paper explores the role and significance of case-based reasoning in design. It commences with a short discourse of what is involved in the design process and specifically in knowledge-based design. It follows by discussing what is entailed in design using compiled knowledge drawing upon the schema developed for design prototypes. The strengths and weaknesses of this approach are discussed. The paper goes on to discuss knowledge-based design using case knowledge. It argues that both specific and generalised knowledge are required for a design system and that what is in a case is best achieved by drawing upon the design prototype schema.

Finally, an argument is put forward for a heterogeneous knowledge-based computer-aided design system which incorporates CAD databases, compiled knowledge and case knowledge. While the examples are drawn from the architectural field, it is suggested that the approach is more general in its application.

Design Knowledge

Design

The role of design is to abduce specific solutions to given problems. This process entails the translation of a design problem given in functional terms to a design description given in structural terms. In most cases this process involves first describing the behaviour required from an artifact since structure/behaviour relationships are easier to determine than function/structure relationships except in well-understood situations. This process of translating goals to means is a process of going from the general to the specific, that is to a defined instance.

The Role of Experience in Design

Function is a human construct and therefore no inherent relationship between function and structure exists. However, the relationships between behaviour and function are more amenable to interpretation through teleological processes. Given a known structure, its behaviour can be interpreted (analysed) using processes as in qualitative physics (Bobrow, 1984; DeKleer and Brown, 1984; Kuipers, 1984). Hence, knowledge about relationships between structure and behaviour and function can be derived and stored. In order to translate function into structure, known structures which relate to that function can be retrieved and manipulated. Design is thus a matter of experience, that is, of acquiring sufficient knowledge about a variety of relevant structures and their behaviours and functions.

Compiled Knowledge and Case Knowledge

Compiled or general knowledge is knowledge obtained from a number of individual experiences. This knowledge may be thought of as rule-like in which a 'rule' exists which is general in nature covering all

the individual experiences but without the specific details which make the experiences individual. This knowledge is obtained usually through an abstraction process although it may be learnt directly. For example, one can learn (from books, lectures, etc) that flashing is required under window sills without necessarily undergoing various undesirable experiences. Although one may learn the reasons why this general knowledge is applicable, the 'rule' itself usually does not carry the explanation. Moreover, this general knowledge does not carry with it any particular solution. A particular situation has to be fitted to a general 'rule' and the general conclusion applied. There needs be knowledge on how to apply the conclusion to any particular situation. For example, the flashing in a cavity brick wall is different to the flashing in a timber-framed wall. Examples of compiled knowledge forms are procedures and formulae as used in computational approaches, rules as found in rule-based expert systems or decision tables and decision trees and schemas such as design prototypes which chunk in one representation several elements of general knowledge.

Specific or case knowledge is knowledge in which the actual experience or episode is stored including all relevant details regarding the situation. When a similar experience is faced, all the relevant parts of previous episodes are retrieved and this knowledge is then applied to the situation at hand. This knowledge includes reasoning and justifications as well as the goals and means. Examples of case knowledge include precedents in law, and specific design examples.

Design Using Compiled Knowledge

Compiled knowledge is knowledge of a general form covering a range of situations. The generalization is divorced from any particular instance which may have formed the generalization and thus from any history of its creation (Riesbeck and Schank, 1989). As such, knowledge of specific situations is not encoded therein although more specific situations may be covered by more specific compiled knowledge and/or computational procedures.

Rules

Rules are examples of compiled knowledge. In knowledge-based systems, rules take the form of

IF <conditions> THEN <consequents>

where some consequents ensue whenever some set of conditions are met. These conditions apply to a class of instances and the consequents may include actions to be carried out as well as factual inferences. Expert systems use such rules usually of a heuristic nature or using 'rules of thumb'. Examples of rule-based expert systems in design are RETWALL (Hutchinson et al, 1987) and SOLAREXPERT (Rosenman and Gero, 1989).

Schemata—Design Prototypes

Schemata are separate modules of general compiled knowledge. They have been described as 'an organized framework of knowledge about the world' (Sowa, 1984). The schema concept has been implemented in artificial intelligence variously as frames (Minsky, 1975; Fikes and Kehler, 1985); schemata (Rumelhart, 1980); and scripts (Schank and Abelson, 1977; Schank, 1982).

Design prototypes are schemata chunking together knowledge regarding a class of design experiences. They contain knowledge about function, behaviour and structure properties of a class of design elements as well as knowledge regarding the relationships among these properties and between the design elements and their context as well as knowledge regarding the relationships between design prototypes. Part of this knowledge includes the necessary procedures for the generation of an instance, i.e. a design solution from a given design problem specification.

Design prototypes are structured in a hierarchical organization from general to specific, with specific design prototypes inheriting properties from the more general. Figure 1 shows an example of a design prototype while Figure 2 shows an example of a hierarchy of design prototypes.

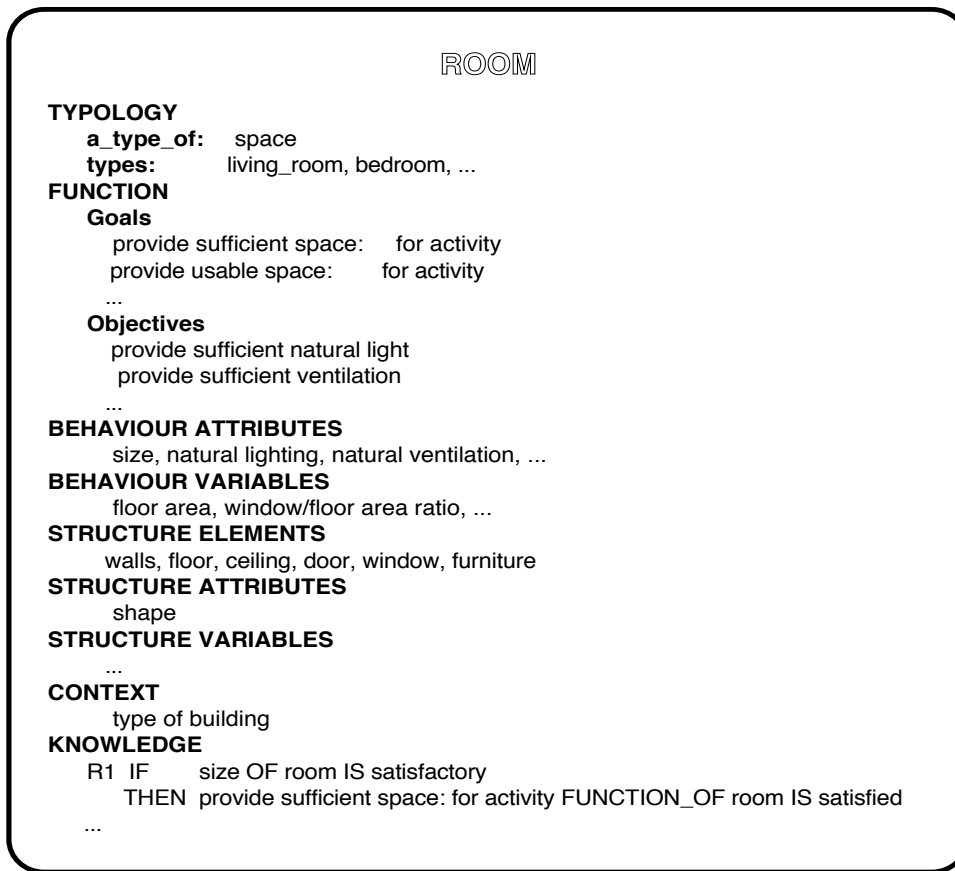


Figure 1. Example of 'room' design prototype

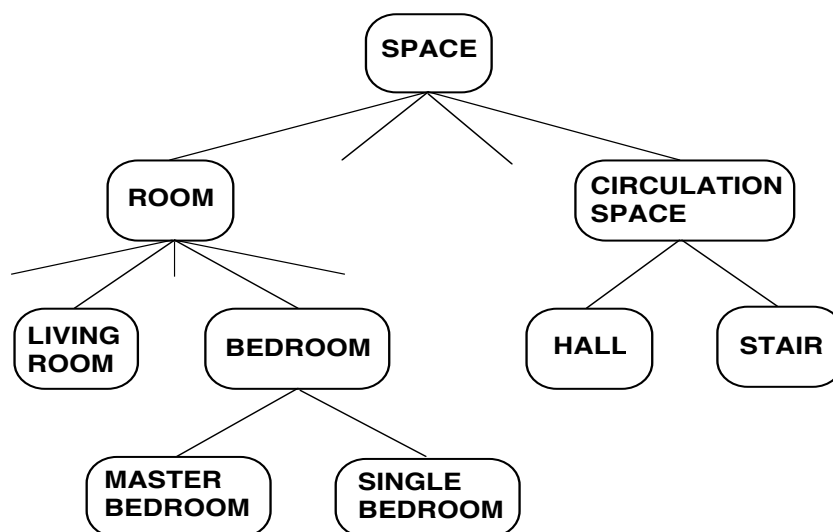


Figure 2. Design prototype hierarchy

Designing with Design Prototypes

Designing with design prototypes entails the generation of a specific instance in response to a given design problem. A design problem is usually given in terms of some function(s) to be satisfied although it may be given in terms of required behavioural performances, constraints regarding the behaviour and/or structure properties or any combination of these. Since design prototypes are structured hierarchically, the process of designing with design prototypes entails the selection of the most specific design prototype, its further specialization by selecting the appropriate structure variables and finally the instantiation of these variables by finding the values which will produce a design satisfying the given requirements.

The knowledge encoded in design prototypes in the form of generative rules supports a refinement schema and can be used efficiently in routine design (Oxman and Gero, 1988; Oxman, 1990) although suggestions have been made as to their use in non-routine design (Rosenman and Gero, 1989). In routine design, the process is as described above. The matching of design prototype to the design requirements is carried out from its function and/or behaviour properties and the description of the design prototype is sufficiently complete to carry out the specialization and instantiation processes. In non-routine design, no single design prototype exists which allows this process and the task will involve such processes as the merging of design prototypes, making analogies to design prototypes in another domain and creating new design prototypes from first principles (Rosenman and Gero, 1989).

Issues Concerning Generalized Knowledge Schemata

Designing with design prototypes, as with all generalized schemata, is computation oriented and repetitive. There is no learning of previous design experiences and the same problem given again is solved by using the same processes as before. Moreover, the fact that compiled knowledge deals in generalizations means that the refinement processes of prototype instantiation cannot always deal with specific situations, which may in fact be exceptions to the general rule.

As with all generalization schemata, there is the difficulty of constructing generalizations. It is not generally accepted, in cognitive psychology for example, what the correct form of a category (generalization) is. There is no universal agreement as to what comprises a category and what model of categorization to use. All attempts at actually constructing generalizations have met with great difficulty except for well-defined and limited domains. No generalization constructed seems to be able to cover all the members of that generalization. In addition there has been much criticism about the applicability of any predetermined system with regards to their use by different designers with different viewpoints.

Design Using Case Knowledge

Case-Based Reasoning and Case-Based Design

Case-based reasoning is a well-defined paradigm in artificial intelligence (Riesbeck and Schank, 1989; Kolodner, 1987; Hammond, 1986). It is based on the premise that humans reason from specific experiences rather than by following a set of general guidelines. For example, reasoning from precedents is one of the basic methodologies in law. Case-based reasoning relates a current situation to the closest most specific experience in memory and uses that experience to solve the problem at hand. It is thus a memory-based approach rather than a computation approach, whereby solutions to problems already solved need only be retrieved rather than computed again. The key factors in case-based reasoning are the storage of cases as complete patterns of experiences including the reasoning process, the ability to be reminded of the most appropriate case and the application of that case to the current situation. Application of the case may either be a direct application if the current situation and that of the case match exactly, or there may be a need for some modification of the case. This modification may be of various degrees of severity. Case-based reasoning uses the strategies of modification and repair to effect such modifications. New cases are thus produced either as variations on the previous case or, in extreme situations, as new cases. Case-based reasoning thus incorporates a learning capacity in the form of new cases being incorporated into a dynamic case base.

Searching for a case is based on indexing cases with regards to various factors, e.g. goals and attributes. The more efficient the indexing, the more efficient the search. Retrieval is a matter of pattern matching, i.e. matching a required pattern of requirements to an existing set. This match may be exact or partial. In the case of partial matches, some criteria are required to determine the 'best' partial match. Matches may be made to parts of several cases and a new case results from combining elements from these cases, if consistency is satisfied.

Case-based reasoning has been applied to various tasks such as classification (Kolodner, 1987), planning (Hammond, 1986; Alterman, 1986), legal reasoning (Bain, 1986), speech recognition (Bradshaw, 1987) and pronunciation (Lehnert, 1987). While there are many similarities between planning and design, planning differs from design in that the problem of planning is essentially that of scheduling a set of actions. In design, the form of the elements themselves is often unknown and the problem involves topological, geometrical and physical properties and relations between them. Hammond (1986) compares the CHEF program to design problems in architecture but the analogy is relevant only to systems buildings where the range and type of available components is well-defined and design involves an additive process of components.

Design Cases

Design cases are specific design episodes of previous design experience serving as examples or precedents for future design problems. They contain an entire design episode, i.e. the problem, the solution and the process for arriving at this solution. The problem contains those requirements that had to be satisfied, including goals and constraints, i.e. function, behaviour and structure as well as any contextual information. The solution contains a description of all the instances of elements necessary as well as their relationships to each other and to the design as a whole. The design process contains information regarding the knowledge, both procedural and declarative, used in making decisions and the justifications for making those decisions. These justifications may include both successes and failures including other alternatives considered but not selected plus information as to what conditions may lead to their selection. This process information is termed the history of the design case. This history is of extreme importance in guiding the adaptation process when modifications to a design case are necessary. Unlike generalized knowledge, wherein a range of solutions is implied, a case contains a description of a design solution in terms of variables instantiated to specific values. However, while the case may contain various instance descriptions, it is, in itself, a holistic description, a pattern of a total experience.

Specific Generalization vs Instance vs Design Case

In a generalized schema, such as the design prototype schema, the hierarchical structure allows for very specific descriptions, see Figure 3. The more specific the design prototype, the more specific its description, especially its structure description. The design prototype, 'vehicle', may have function and behaviour information but can have little specific structure description, whereas the design prototype, 'beetle', has much more specific structure description. However, some information is still be general so as to be able to produce a number of instances of specific 'beetles'. As such, those behaviour properties associated with those general structure properties will also be general or have expected values. An instance of a 'beetle' would have all its variables instantiated to a specific value, i.e. colour red, cloth seat covering, with air-conditioning, etc. The instance of an artefact includes all instances of elements forming part of that artefact instance, see Figure 4. An instance will have its behaviour properties instantiated to actual values or these can be derived from the structure. These actual values can be compared to the expected values and the performance of the instance evaluated. An instance may be derived from a design prototype through refinement in a top-down fashion or, alternatively, an instance may be described unrelated to any such generalization.

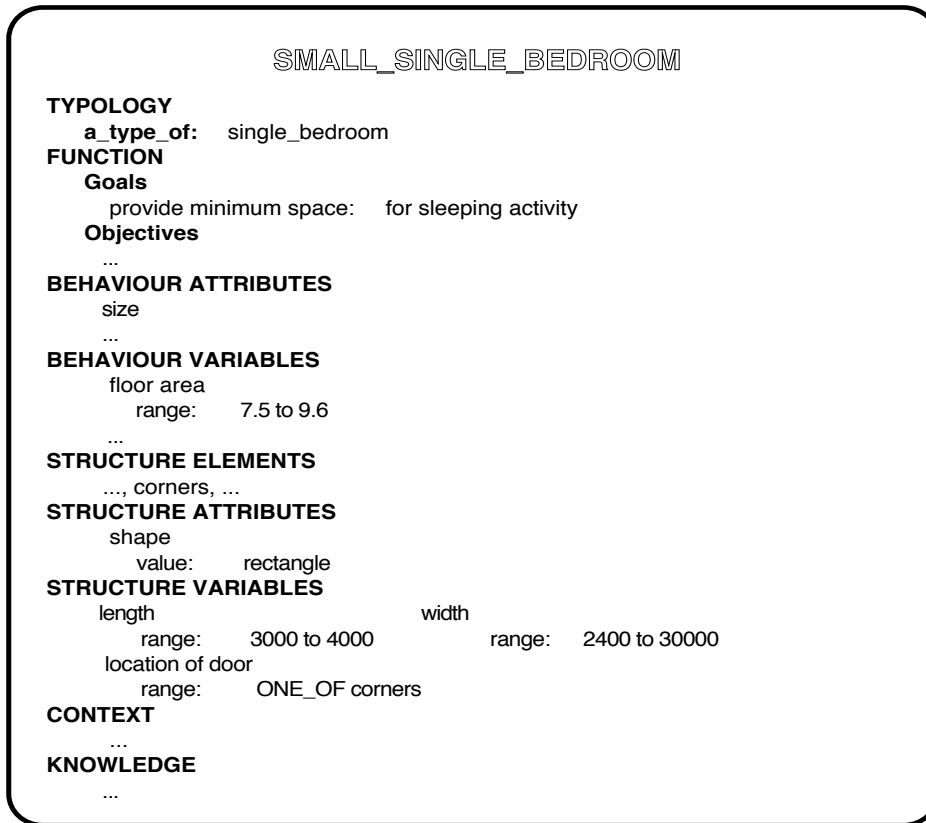


Figure 3. Example of Small_Single_Bedroom design prototype

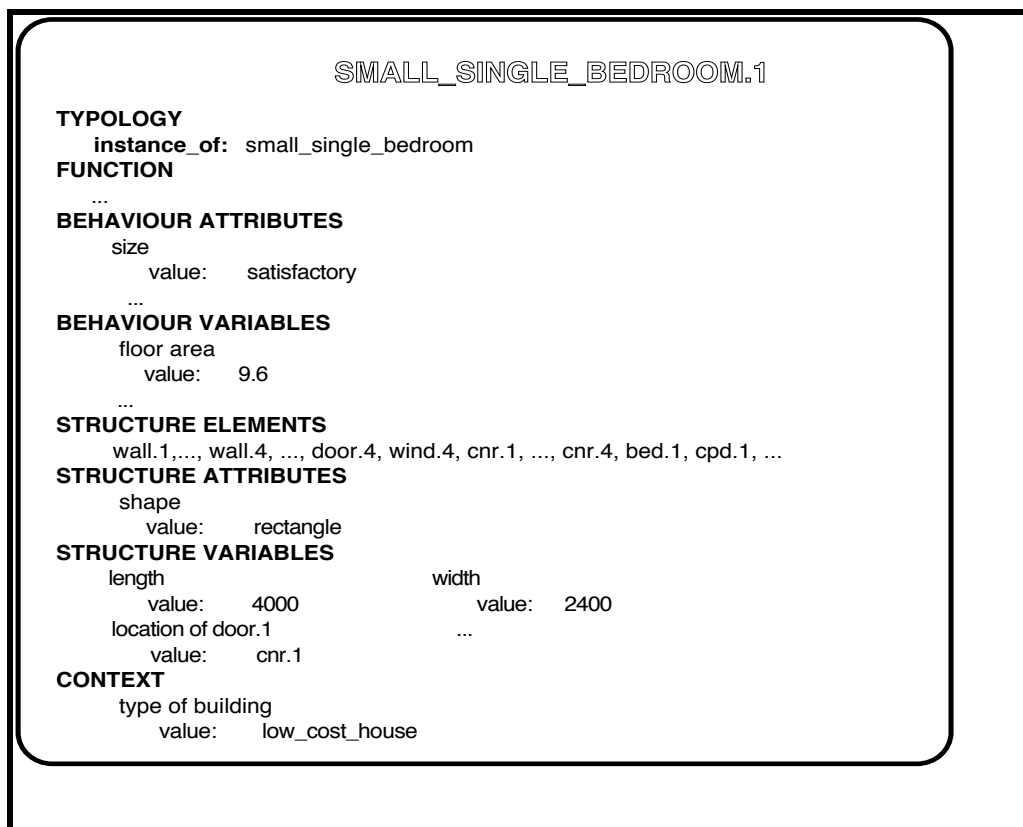


Figure 4. Example of Small_Single_Bedroom instance

Like a design prototype and unlike an instance, a design case is a total description of design experience. However, a design case differs from a set of design prototypes in that its structure description, i.e. its design solution description, is specific as for a set of instances. What differentiates a design case from design instances is that a design case contains a history of the design experience, i.e. it is not merely a description of the artefact itself but also of the problem, processes, and justification for decisions taken. An example of a design case with its constituent instances is given in Figure 5. Since a design case is the encoding of a particular design along with its salient aspects of its episodic design history, it serves as an exemplary mapping maintaining the semantics of its creation. A unique design case might be recorded as the successful design response to an unusual requirement or a particular constraint.

Designing with Design Cases

Designing with design cases involves the use of a previous like experience to solve a given design problem. As in designing with design prototypes, the closest matching experience must be found, however unlike designing with design prototypes, no specialization or instantiation is involved. The design solution exists as a stored description and, if the design situation is sufficiently similar, can be directly appropriated. An example of this, is the use of standard details in a design office, or even 'standard' designs for building types. In some cases, minor modifications may be necessary, e.g. some change in dimensions while in other cases more major modifications may be necessary, requiring some redesign. The processes involved in case-based designing are problem anticipation, search, match, retrieve, select, modify, repair and store.

Problem anticipation. It is usual for design problems to be given with incomplete information. As in the example of menu planning by Kolodner (1987), the problem specification does not mention the possibility of some of the guests being vegetarian. Since this problem had occurred previously (and having led to a failure of the plan was noted by the system) the system, JULIA, asks the planner if there are any vegetarian guests. Similarly, in design problems, in order to avoid inappropriate design cases being accessed through lack of initial information, it is desirable for a system to try and ascertain as much relevant information as possible.

Search. Given a problem description of requirements including functions to be achieved, required behaviour performances, the design environment and even constraints on values of structure variables, the case base must be searched to find an appropriate design case. The utility of case-based designing is strongly dependent on the efficiency of the search procedure. Searching could be sequential, parallel or direct using an indexing mechanism. Indexing must be done on the function, behaviour, structure and context features.

Match. An appropriate case for consideration is found with regards to the matching of above mentioned features. Perfect matching, i.e. where the required features are found exactly in a case, is unusual. Partial matching occurs when some of the features are matched or the features are matched to some degree.

Retrieve. A case which matches to some defined degree needs to be retrieved for consideration. This may or may not involve display of these cases to the users for perusal and consideration.

Select. A selection of a single case as the basis for determining the design solution has to be made. Alternatively, if only part of a design case is required, then several design cases may require to be selected, and the necessary parts of each extracted. In either situation, the 'best' matching design case should be selected. Selection of the 'best' design case can be on the basis of the most similar or the most useful match (Kolodner, 1989). Selection can be carried out by the system or by users after consideration of an appropriate set of candidates retrieved by the system. Selection by the system based on partial matching entails such factors as the importance of the features matched as well as how close they are matched. A similar problem has been addressed in the specialization of design prototypes (Rosenman, 1990).

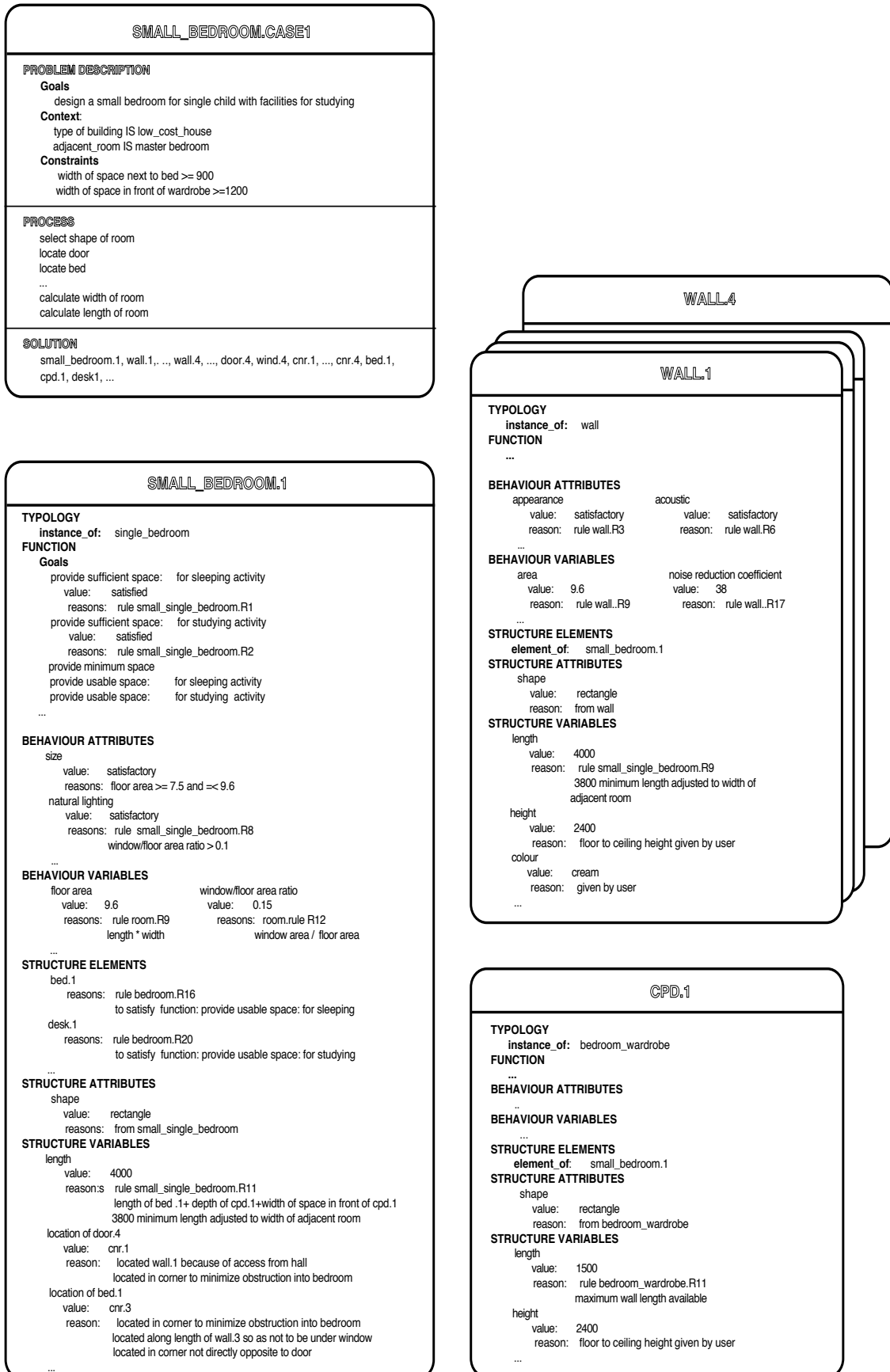


Figure 5. Example of Small_Single_Bedroom design case showing related instances with their history

Modify. Where a design case is selected which does not match the design requirements sufficiently, some modifications will be necessary. This may involve the replacement of variables with other variables or simply the alteration of some values of variables.

Repair. In many situations, a modification to an existing design case based on substitution of variables or modification of values will cause some performance failure in some other behaviour or function. For example, decreasing the cross-sectional area of a column to satisfy some new spatial requirement may cause buckling. Other modifications may be considered but none may be satisfactory. One of two directions now needs to be taken. Either an alternative design case is selected based on the new information known regarding the necessity for modifications and the effects of modifications or the current selected design case is modified in such a way as to make it acceptable. This latter process is known as the process of repair in case-based reasoning.

Store. After a design case has been modified or repaired, a new design case has been generated. If this new design case is considered to be sufficiently important as a design experience different to existing design cases, then it must be stored in the case base with appropriate indexing. Where the failure of solutions is seen as an important piece of information to the anticipation of future problems, this must be noted in the design case.

What's in a Design Case?

So what is in a design case? Firstly, a description of the design problem including the noting of critical problem areas which need to be anticipated. Secondly, a description of the design solution in terms of structure, actual behaviour and functions achieved and thirdly, and most importantly the design history.

Issues Concerned with Case-Based Design

Number of design cases

The first issue is the question of when is a design experience sufficiently relevant or important to become a design case. That is, what makes an episode worthwhile to be stored as a design case. What degree of difference is required between design cases to make it worthwhile to store rather than to derive from another design case through modification. This is obviously related to the degree of difficulty of the modification but more importantly to a distinct 'lesson' that a design case serves. That is, each design case must help to solve some distinct problem so as to serve as an example of likely types of design problems. The differences between the various design cases should not be too great or else there will be a great deal of modification and repair necessary and there will be little advantage over a generalized design approach. Yet the difference cannot be so small as to be too trivial and as a consequence the size of the case base will be so enormous and detailed that the search and match procedures become extremely inefficient.

Description of case

What features are relevant for a design case? What should the organization of such features be? How can a feature in one design case be related to another feature in a different design case which are related in an abstract manner? For example, if a description of one design case involves the description of a chair that has legs whereas another has a description of a chair with a pedestal, how is the relation between these legs and the pedestal made? How can these questions be resolved without recourse to a generalization, such as 'chair support'?

Organization of case base

In a large case base, there will be many design cases where a large number of features will be the same. For example, in the description of design cases dealing with chairs, many features, such as the chair seat, seat back may be the same in many cases, with some other varying features, such as armrests, legs, pedestals. For efficiency's sake, rather than storing every design case individually in its entirety, a hierarchical approach may be required. In case-based reasoning, Schank's theory of dynamic memory

(Schank, 1982) provides an organization with knowledge structures called Memory Organization Packets (MOPs). Kolodner's CYRUS system (Kolodner, 1984) was the first implemented case-based reasoning system founded on Schank's MOPs. Kolodner's memory organization uses structures called Episodic Memory Organization Packets (E-MOPs). E-MOPs index similar episodes by their differences.

Generalizing from cases

Although there may exist knowledge relating one episode to another, case knowledge is specific to the case or episode. For example, a case may describe a window detail of a some aluminium sliding window on a brick-on-edge sill in a brick cavity wall. If the new situation is to design the same aluminium window in a timber-framed wall the knowledge regarding this situation does not reside in the case. When a different situation is faced, yet a similarity is found to an existing case recourse must be made to some generalization process to apply the lessons of one situation to the new situation.

Modification knowledge

Only some very limited and specific knowledge regarding modification and repair of a design case can be in the design case itself, since future problems cannot be anticipated. In order to take care of a wide range of possible situations this type of knowledge must be of a general nature. Also there needs to be some evaluation of modification strategy, such as, is it better to modify or in some situations is it better to start from a new design case.

Integration of Generalized and Specific Knowledge

The design prototype and the design case represent two different approaches to knowledge-based design. Design prototypes are efficient in their process of design generation and refinement. Design cases are powerful because of their richness of specific detail and history. Case-based design is founded on the use of memory to retrieve specific experiences and utilize specific detailed solutions when the direct experiences are sufficiently close to be of use. However, case-based design lacks the power of generalized knowledge to deal with situations somewhat removed from a specific experience. Case-based design can potentially supplement design prototype-based design with its retrieval, adaptation and repair facilities. An integration of both approaches would use the strengths of both. That is, when direct experience is available it can be used, otherwise recourse can be made to general principles. That is, case-based design can be used when a suitable design case exists, otherwise recourse can be made to the generating functions of a suitable design prototype.

A system founded on design cases alone means that each case is a unique representation of a total design experience. A design case dealing with the design of a bedroom may be separate to one dealing with a dining room. Yet many problems, decisions and solutions of the two rooms, will be similar, if not the same. For example, the need for natural ventilation, natural light and for access, will result in the inclusion of a window and a door, not to mention elements such as walls, ceilings and floors. The use of a generalized schema allows for the abstraction of like experiences allowing for the formation of higher level concepts, such as 'room' in the above example. This allows for relationships between features in design cases based on a higher level concept rather than on some list of properties. Moreover, basing case descriptions on the representational formalism of the design prototype, answers the 'chicken and egg' question of how to extract relevant features for a design case and where generalizations come from, since both are formulated with respect to each other. In such an approach, general knowledge resides in the design prototype and the design case is seen as providing detailed specific knowledge regarding a situation or an exception to the general 'rule'. The design case contains the problem description, the process used and the set of instances which constitute the solution. These instances contain the information as to which design prototypes and which sources of knowledge were used (and why) to derive attributes and values.

The integration of design cases with the design prototypes representation allows for the modification and repair of a design case through the generating and instantiating capabilities of the design prototype. If a design case retrieved is not identical to the required situation modification may be required involving a change in the value of some variables, e.g. the length of a room. Because a design case encodes its history, including which design prototypes and knowledge are associated with the derivation of values, the respective design prototype can be accessed and the respective knowledge re-run to provide a new instantiation through value modification.

When a newly derived value is unsuccessful in adaptation (and no such modification is successful), then new variables must be generated. Repair involves a change in the variables. This requires access to a higher level abstraction, namely the design prototype, for the construction of a new structure to satisfy behavioral and functional requirements.

This integration of the design prototype and design case formalisms allows for the generation of design cases from design prototypes. Figure 6 shows the relationship between design prototypes, instances generated from design prototypes and design cases.

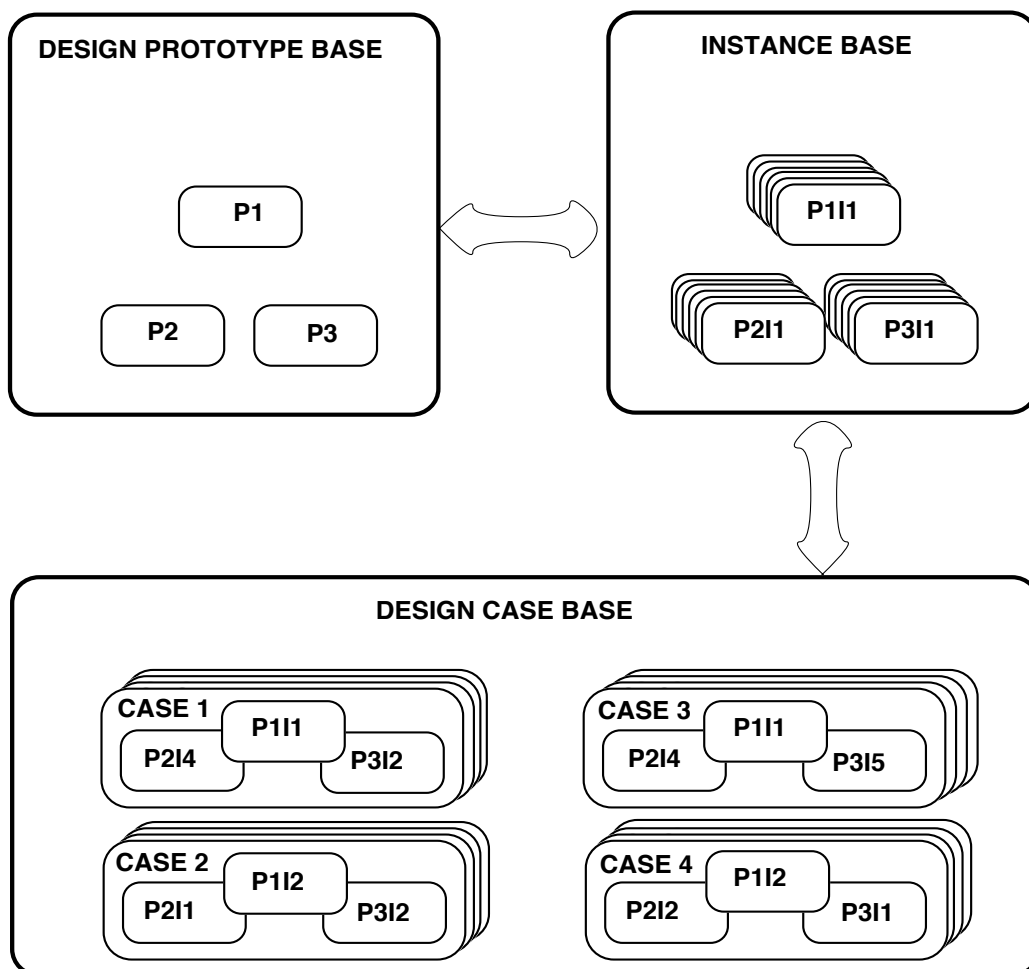


Figure 6. Relationship between design prototypes, instances and design cases

There are three different types of design cases which may be generated in this way. The first type is the design case in which all the variables and their values are derived through the generating process of the design prototype system. The second type is the design case in which the user selects some of the values of the variables and the third type is the design case in which the user selects some of the variables as well as some of their values. The first type occurs when a defined situation can be matched exactly by a

design prototype and all variables and their values are catered for. Re-running the same situation would result in exactly the same set of instances being generated. With the second and third types, re-running the same situation may not result in exactly the same instances being generated, as users may vary their selection of variables and values.

Computer-Aided Design

CAD and CAD Databases

Computer-aided drawing (CAD) systems use databases which solely represent structure properties and traditionally only topology and geometry although other properties, such as material, may be described. Depending on the system, the representation may be of purely graphic elements, i.e. points and lines, or two- and three-dimensional shapes, such as rectangles, circles, prisms, spheres. Labels may be attached to such shapes so that these shapes can be referred to as objects with given properties. Basically, such databases describe instances of such objects although they may also have the ability to create a class descriptions of objects which are stored in object libraries.

Integration of CAD Systems with Knowledge Bases

Lately, there have been moves toward the integration of generalized knowledge bases with graphic systems using CAD databases. Such knowledge bases provide the means of relating semantic information to the purely syntactical information in the CAD databases. These knowledge bases take the form of rule bases of expert systems or hybrid forms of rules and frames. More recently, there has been work on the integration of design prototypes as the general domain knowledge to enhance the integration of such CAD databases with expert systems which deal with specific application knowledge, such as design codes (Balachandran et al, 1990). The further addition of case knowledge will increase the ability of design systems to provide support for designers.

In the situation of generating designs, a design problem may be related to a previous design case, rather than to the general domain knowledge. This design case would also include that information related to a CAD database description and would enable the design artefact to be visualized or perhaps, sent to some application program for further evaluation. Modification of the artefact after visual evaluation, by graphic manipulation or direct modification of the database could serve to generate a new design case. In the situation of design evaluation, a design modelled graphically could relate the information thus given in its database to that of an existing design case and certain conclusions drawn that may not be available in the general domain knowledge. For example, building regulations may have general provisions to the effect that a single-dwelling house may not have a rise of more than two storeys except in special circumstances as allowed by Council. The general knowledge does not know, indeed cannot know in advance, all the particular set of circumstances which may lead to Council permitting a particular three-storey house design. However, there may exist a specific design case in a situation similar to the current one where a three-storey house was permitted. The matching of the designed house and the design case will be done through the information in the database plus other specific descriptive information as required. In such examples, a specific design precedent is used rather than a general set of rules which cannot cover every possible situation.

Conclusions

This paper has argued that both generalized knowledge, in the form of design prototypes, and specific knowledge, in the form of cases are valid and important sources of design knowledge. They, not only need not be used exclusive of each other but rather, should be unified into an integrated system so as to utilize the strengths of each. An important result of such an integration is the structure imposed on the design case by the design prototype formalism as well as the ability to access higher level abstractions where necessary.

While, at this stage the focus of this paper is towards routine design, some aspects of non-routine design can be approached using case-based design especially since the processes of modification and repair correspond to those of adaptation or mutation. However, since case-based design is founded on the use of specific experiences it is difficult to see how designs that are new in kind can be generated without recourse to generalizations.

Acknowledgments. This work is supported by a continuing grant from the Australian Research Council.

References

- Alterman, R. 1986. "An adaptive planner." *Proceedings of AAAI-86* 1: 65-69.
- Bain, W. M. 1986. "A case-based reasoning system for subjective assessment." *Proceedings of the National Conference on Artificial Intelligence*, pp.523-527.
- Balachandran, M., M. A. Rosenman, and J. S. Gero 1990. "A knowledge-based approach to the automatic verification of designs from CAD databases." *Working Paper*. Design Computing Unit, Department of Architectural and Design Science, University of Sydney, Sydney.
- Bobrow, D. G. 1984. "Qualitative reasoning about physical systems: an introduction." *Artificial Intelligence* 24: 1-15.
- Bradshaw, G. 1987. "Learning about speech sounds: the NEXUS project." *Proceedings of the Fourth International Workshop on Machine Learning*, pp.1-11.
- DeKleer, J. and J. S. Brown 1984. "A qualitative physics based on confluences." *Artificial Intelligence* 24: 7-83.
- Fikes, R. and T. Kehler 1985. "The role of frame-based representation in reasoning." *Communications of the ACM* 28, no. 9:904-920.
- Gero, J. S. 1987. "Prototypes: a new schema for knowledge-based design." *Working Paper*. Architectural Computing Unit, Department of Architectural Science, University of Sydney.
- Hammond, K. J. 1986. "CHEF: A model of case-based planning." *Proceedings of AAAI-86* 1: 267-271.
- Hutchinson, P. J., M. A. Rosenman and J. S. Gero 1987. "RETWALL: an expert system for the selection of and preliminary design of earth retaining structures." *Knowledge-Based Systems* 1, no. 1:11-23.
- Kolodner, J. L. 1984. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*. Lawrence Erlbaum, Hillsdale, New Jersey.
- Kolodner, J. L. 1987. "Extending problem solver capabilities through case-based inference." *Proceedings of the Fourth International Workshop on Machine Learning*, pp.167-178.
- Kolodner, J. L. 1989. "Judging which is the "best" case for a case-based reasoner." *Proceedings of Workshop on Case-Based Reasoning*, pp.77-81.
- Kuipers, B. 1984. "Commonsense reasoning about causality: deriving behaviour from structure." *Artificial Intelligence* 24: 169-203.
- Lehnert, W. 1987. "Case-based problem solving with a large knowledge base of learned cases." *Proceedings of the National Conference on Artificial Intelligence*, pp.301-306.
- Minsky, M. 1975. "A framework for representing knowledge." In P. H. Winston (ed.), *The Psychology of Computer-Vision*. New York, McGraw-Hill, pp. 211-277.
- Oxman, R. E. 1990. "Design shells: a formalism for prototype refinement in knowledge-based design systems." *Artificial Intelligence in Engineering* 5, vol. 9: 2-8.
- Oxman, R. E. and J. S. Gero 1988. "Designing by prototype refinement in architecture." In J. S. Gero (ed.), *Artificial Intelligence in Engineering V: Design*. Elsevier/CMP, Amsterdam, pp.395-412.
- Riesbeck, C. K. and R. C. Schank 1989. *Inside Case-based Reasoning*. Lawrence Erlbaum, Hillsdale, New Jersey.
- Rosenman, M. A. 1990. "A qualitative evaluation approach to the specialization of design prototypes." *Working Paper*. Design Computing Unit, Department of Architectural and Design Science, University of Sydney, Sydney.

Rosenman, M. A. and J. S. Gero 1989. "Creativity in design using a prototype approach." *Preprints Modelling Creativity and Knowledge-based Creative Design*. Design Computing Unit, Department of Architectural and Design Science, University of Sydney, pp.207-232.

Rosenman, M. A. and J. S. Gero 1989. "SOLAREXPERT—an expert system for evaluating passive solar energy designs." In B.H.V. Topping (ed.), *Artificial Intelligence Techniques and Applications for Civil and Structural Engineers*. Civil-Comp Publications, Edinburgh, pp.131-139.

Rumelhart, D. E. 1980. "The building blocks of cognition." In R. J. Spiro and W. F. Brewer (eds), *Theoretical Issues in Reading Comprehension*. Lawrence Erlbaum, Hillsdale, New Jersey.

Schank, R. C. 1982. *Dynamic Memory: A theory of Reminding and Learning in Computers and People*. Cambridge University Press.

Schank, R. C. and R. P. Abelson 1977. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum, Hillsdale, New York.

Sowa, J. F. 1984. *Conceptual Structures*, Addison-Wesley, Reading, Massachusetts.

This is a copy of the paper: Rosenman, M. A., Gero, J. S. and Oxman, R. (1991). What's in a case: the use of case bases, knowledge bases and databases in design, in G. N. Schmitt (ed.), *CAAD Futures '91*, ETH, Zurich, pp. 263-277.