Artificial Intelligence in Design- AID '94, J. Gero and F. Sudweeks (eds.), Kluwer Academic Publishers, Doordrecht, The Netherlands, 1994, pages: 145-162.

CASE BASED DESIGN IN ARCHITECTURE

B. DAVE and G. SCHMITT
Swiss Federal Institute of Technology, Zurich

and

B. FALTINGS and I. SMITH
Swiss Federal Institute of Technology, Lausanne

Abstract. Computational support in the domain of building design is hampered by the need to control generation and search processes both of which are elusive due to the lack of strong domain theories. Case based reasoning paradigm may be useful to overcome some of these difficulties. A case based design system is presented here that enables case adaptation and case combination of design cases to generate new design solutions more efficiently. Some issues in our approach that are different from other projects with similar aims are also discussed.

1. Introduction

Case based design (CBD) systems are being investigated in many domains as a solution to overcome the complexity of design generation and search processes as well as to get around issues that arise from a lack of strong domain theories. The domain of building design exhibits both these characteristics and thus, application of case based design systems in this domain is of increasing interest.

The approach that we have taken in our work has much in common with some of the key issues that are typically associated with case based reasoning (CBR) systems in general. At the same time, we also believe that the domain of building design has some special characteristics that either enable or require us to reframe some issues in CBR in a different way.

In this paper, we first present (Section 2) some characteristic uses of cases in building design that distinguish them from other domains. This is followed by a description of a computational framework (Section 3) under development that supports reuse of design cases for generation of new designs by the processes of case adaptation and case combination. Using the observations presented in Section 2 and the results presented in Section 3, a broader discussion of our approach and open issues in CBD are presented in Section 4.

2. Design Cases in Architecture

The domain of architectural design has maintained- sometimes an uneasy-existence between the sciences and the arts. It is a discipline that is called upon to express simultaneously universals and particulars. As a result, articulation of a theory of architecture is constantly in a state of flux. Given this state of affairs, education in architectural design relies heavily upon the use of cases as a vehicle of discourse between the teachers and the students; the hope being that the particulars in a given case offer a holistic view of design issues that are difficult to articulate or view if they were taken up separately. This mode of example-based teaching and learning is perhaps not unique to the discipline of architecture, which entails developing a facility for making generalisations as a function of new examples that are encountered. The dialectical process of using past knowledge to solve new design problems also continues into professional practise. In our work, we focus on reusing one or more particular case in response to a new design problem; more specifically, our intention is to develop an experimental computational framework within which we can propose, test and validate theoretical issues pertaining to case based reasoning in architectural design.

There are two approaches in architectural design that are of interest to us, namely case adaptation and case combination. In the following, we cite two examples of both these approaches. The design for the architecture school at the University of Houston is a direct adaptation of a 1773 design sketch for a "house of education" by the French architect Claude Nicholas Ledoux. This is an extreme form of case adaptation in which a few details of the original case have been modified but the resemblance with the original sketch has been maintained to a large degree. The adaptation appeals to the references

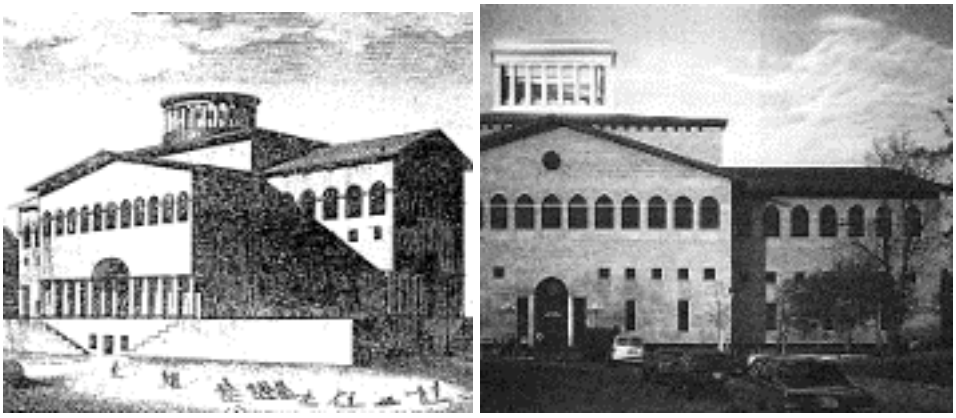

Fig. 1. Left: Sketch for a "house of education", Ledoux, Right: Design for the architecture school, University of Houston, Burgee-Johnson and Morris-Aubry

contained in the original sketch, e.g. simple massing of volumes, that are used as devices to signify some abstract architectural notions, e.g. symmetry has been often associated with wholeness and stability. The important thing to note is that by preserving the most important relations between the massing volumes and their relative proportions, the new design even while incorporating new elements seems to stay quite close to its seed ideas.

Examples of designs that combine elements from many different design cases are available in the *colonial* architectural designs. A prime example of such an approach can be discerned in the works of architect Sir Edwin Lutyens, who left his mark on the British Raj architecture in New Delhi, India. In the design for the Viceroy's House, for example, a number of traditional Indian motifs are combined with the Western classical elements. The product of such combination of partial design cases as developed in Lutyens' designs has been lauded as one of most impressive and innovative designs. Other examples of architectural designs that expressly combine various cases can be seen in many building reuse and extension projects in which typically an existing building is modified to permit a usage that is different than its previous use.

These examples point out some key characteristics of the uses of cases in design, both in terms of why and how cases may be reused in design. Cases may be adapted or combined in a design not only for the efficiency obtained in generation and search processes in design development. Cases may be used not only for filling gaps in a weak domain theory. As the examples cited above show, cases may be used as legitimate instruments and an end in themselves in design activity. The examples also suggest that two of the most common ways in which previous designs may be reused in new contexts are

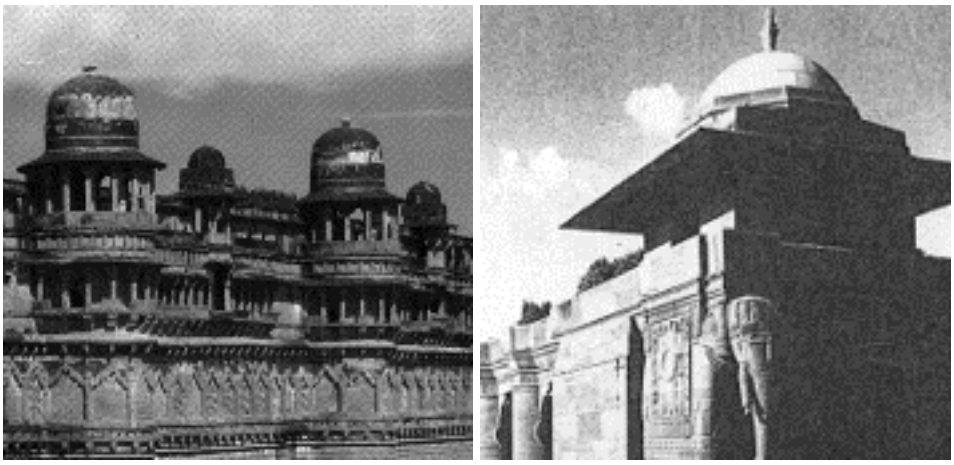

Fig. 2. Left: Traditional Indian chhatri. Right: Guard House, Viceroy's Court, New Delhi, Lutyens

by way of dimensional adaptation and by reusing design elements in new combinations to serve different uses in new design contexts.

3. Computational Framework

To investigate and operationalise the use of cases in design tasks, we have been working on the implementation of a case based design system. A number of references [Hua et al., 1992; Schmitt et al., 1993] describe our work and the implementation of computing ideas, particularly in support of case adaptation. In the following, we briefly recap the overall system framework under development and report on the process of case combination in our system. The presentation is organised in three subsections: case base, case adaptation, and case combination; the fourth subsection describes the current and planned work in near future. The system under development is intended to work together with a designer, thus each subsection describes the information that a user or a designer provides and how the system processes user specifications to generate new information states including design solutions.

Many of the details presented in the following have been implemented and tested using cases that are small as illustrated in various subsections. Once small test cases show promising results, we intend to apply the developed techniques to big and complex building design problems (see Section 3.4).

3.1. CASE BASE

A designer works with a collection of design cases stored in the system. Each case in the case base is a complete three-dimensional geometric model of a design that has been associated with an ordered symbolic information. Thus, for example, a set of *vectors* may represent a *wall* of certain dimensions which, in turn, may be associated with a *space* labelled *living room*. Thus there is a degree of hierarchy (that is conceptually meaningful) in the information represented as well as a certain latitude in storing some information vs. computing it. The 3D geometric model is represented using a CAD modeler which acts as a graphics engine. The CAD modeler also lets us directly manipulate geometric information that it stores and associate additional data structures with any of the data items in the model. Symbolic information that we require is represented as association lists that are attached to relevant graphic entities.

For each case, there are two related abstractions that are maintained: architectural and structural. Spaces and elements that contribute to architectural issues such as functions, access, adjacency, are part of the

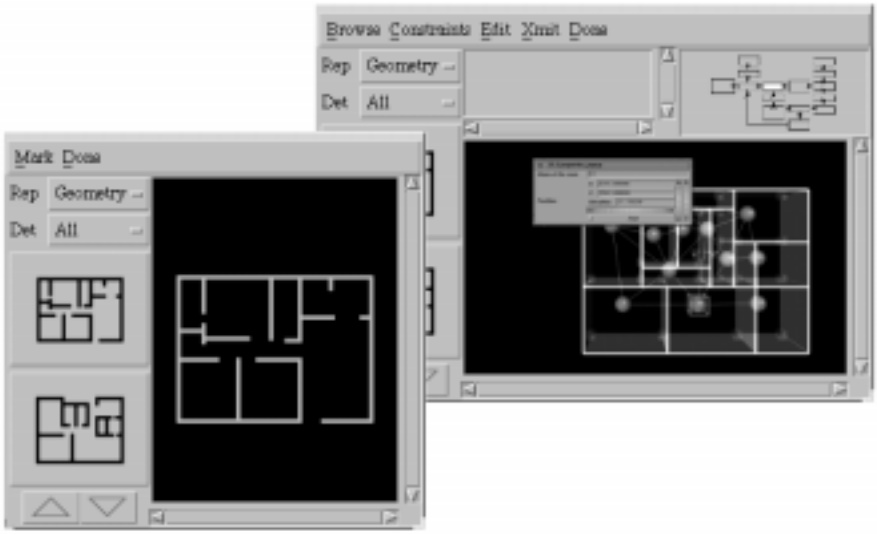

Fig. 3. Case browser and work windows

architectural abstractions. Structural abstractions involve engineering concepts such as structural stability and serviceability. For both these abstractions, there is a defined symbolic vocabulary, and each abstraction is related to the geometric model which provides a common link between various abstractions. Abstractions like these enable us to take advantage of formalised bodies of knowledge such as in structural design or enumerating rectangular layouts.

For a given design problem, a designer browses through the case base and selects one case for adaptation or several cases for combination. For each stored design case, the browser provides a graphic thumb-nail view as well as various representations that are useful for designers, together with a set of defined operations on each representation. The various representations may be *geometry*-based, *topology*-based or *grammar*-based (Figure 3) and they may have been stored with the cases in the case base or may be computed on demand. As mentioned before, all the cases have a geometric representation that is enhanced with symbolic information. Using this information, a set of operations computes an alternate topology-based representation such as a graph encoding adjacency relations between spaces. Finally, in order to encode and operate on some recurring design concepts, e.g., *courtyard*, *symmetric plan* configuration, it may be useful to employ a shape grammar-based formalism that is either explicitly represented or computed on demand.

Further, a particular representation of a case can be also viewed in one of the four possible levels of detail. *All details* shows all available information

about a case; *aggregate* (or zones) shows the explicitly represented or computed aggregation of spaces, e.g. all spaces that are public, private or related to circulation; *center-lines* shows the underlying geometry of spaces without showing the actual elements such as walls and columns; and *bounding box* displays the smallest bounding volume for a given case.

Both, the kinds of representations and amount of detail to display for a given representation, have been selected on the basis of what we find to be most useful for design tasks during various stages in design. For example, while specifying constraints on adjacency or orientation of various spaces, a topological graph with edges and nodes of all spaces may be sufficient. To specify alignment of space separating elements such as walls, only the center lines may be sufficient. On the other hand, it may be useful to treat service cores and areas as an aggregate unit that has to be located in relation to other spaces but the location of individual elements within a service core, such as a staircase and an elevator, may be turned into a separate problem and be treated as such. All the operations, i.e., switching between various representations of a case, between various levels of detail, and interactive, direct manipulation viewing operations, are supported in 3D space.

3.2. CASE ADAPTATION

A solution to a specific design problem has to be developed by adapting the selected case. To this end, the user needs to first specify the new problem context which, in the case of architectural designs, is usually a site description and any programmatic requirements. The site description is input using the same module that is used for case representation, and typically needs specification of a geometric model of site boundaries, its location, orientation, and any other context specific information such as roads that feed the site. Next, the case that has been selected for adaptation is inserted by the user into the new site.

Any new problem specific requirements that the new design problem may require, for example, three bedrooms instead of two in the selected case, need to be specified. These specifications are accomplished by either directly querying and modifying the existing information in the selected case. For example, a node representing a bedroom may be clicked which brings up a property sheet into which desired values such minimum or maximum area may be specified. Alternatively, one may use instances of templates such as a new node that represents a new bedroom to be added; any dimensional or topological constraints are then interactively specified in the property sheet or graphically by dragging edges or nodes in a topological representation. It is important to note that interactive specification of requirements by the user

at this stage indicates what the system will attempt to achieve in a design solution using case adaptation in the subsequent stages of processing.

The result of these specifications is a set of constraints that is derived on the basis of discrepancies between the selected case in its original context and after its insertion into the new design context. This leads to a parameterization of relevant parts of the case in order to eliminate discrepancies. The parameters thus obtained are subject to either *dimensional* or *topological* modification. Dimensional adaptation modifies dimensions of various design elements without introducing any changes in the number of design elements that are present. Topological adaptation modifies the design elements and topological relations between them.

Dimensional modification is carried out by a process called *dimensionality reduction* (Figure 4) which ensures that any constraints that are currently satisfied are not subsequently violated by modification of their relevant parameters. This is achieved by identification of only the subspace of parameters defined by the constraints to be modified. The process leads to an adaptation parameterization that is subsequently solved to find a feasible set of values for all parameters involved. In essence, this process is aimed at solving cycles in a constraint network so that subsequent adaptation procedures are simplified. This process also ensures that any modification introduced by way of adaptation will not create any contradiction between different abstractions of a design. Since dimensionality reduction applies only to equalities, the system subsequently also verifies inequalities. Inequality parameters are handled in two ways: those that are critical are turned into equalities and to which dimensionality reduction process applies.

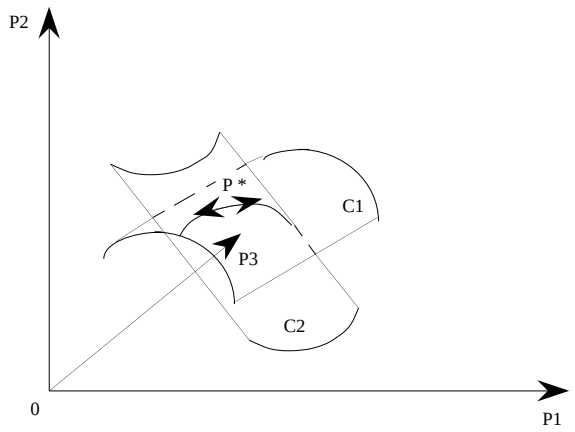

Fig. 4. Dimensionality reduction. Pn and Cn represent parameters and constraints for different case abstractions. P* represents a new parameter which is located on the intersection curve of constraints. By selecting a suitable value of P* along this curve, large and often iterative search is eliminated.

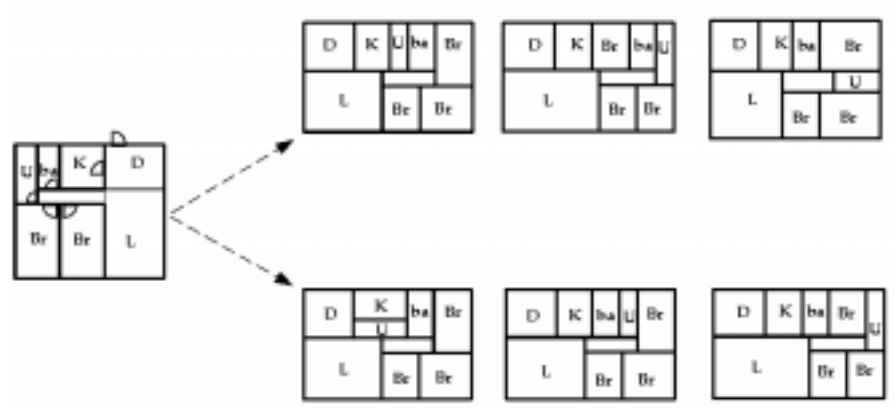

Fig. 5. Left: Selected design case. Right: Design solutions after case adaptation

Non-critical inequality parameters are handled by a constraint propagation mechanism. A detailed discussion of the dimensionality reduction process has been presented previously (Hua et al., 1992). In the event that all the constraints are resolved, one or more design solutions are obtained and displayed. This is the final outcome of a successful case adaptation process; Figure 5 shows the results of one such design problem.

If all the constraints cannot be satisfied then a designer has two choices: either to modify the set of constraints or to select a new case for adaptation. While the selection of a new case involves another cycle of the process described above, modification of constraints may involve topological adaptation which currently requires the user to specify either addition, suppression or rearrangement of parts in the case. Since any proposed topological change in one abstraction is likely to affect and influence another abstraction of the design, proposed or posted topological modifications need to be followed by a new cycle of dimensional adaptation procedure and a check for consistency between all abstractions. If the system fails, another topological modification cycle may be necessary or a new case may need to be selected for adaptation.

For topological adaptation, we have investigated the possibility of using string grammars (Shih and Schmitt, 1993) for (semi)automatic exploration of possible changes. The primary idea in using grammars is to take a selected design case, find its most characteristic elements, and parse this information in another similar combination of elements which preserves relations between elements as much as possible. This is similar to writing computer programs which are parsed into other programs by compilers, which are finally translated into executable code. Devising appropriate topological modifications is a non-trivial phase in the design process that has so far defied a general automated solution since the process relies upon domain

specific and problem specific knowledge. This phase is again followed by a dimensionality adaptation to ensure that the new topology meets the dimensional requirements.

3.3. CASE COMBINATION

Case combination employs the same techniques as case adaptation, only in a slightly more extended and involved fashion. The initial steps in this process, namely specification of a new site context, selection of cases or their fragments that are of interest and any new problem specific requirements that need to be satisfied in the combined design solution, are almost identical to the one described previously for the case adaptation procedures.

The major difference from the user point of view is that two or more cases need to be selected from the case base and areas of interest in each case need to be specified (shown as grey areas in Figure 6). Initially, the system treats the selected case fragments as only the visible part of the case; the system

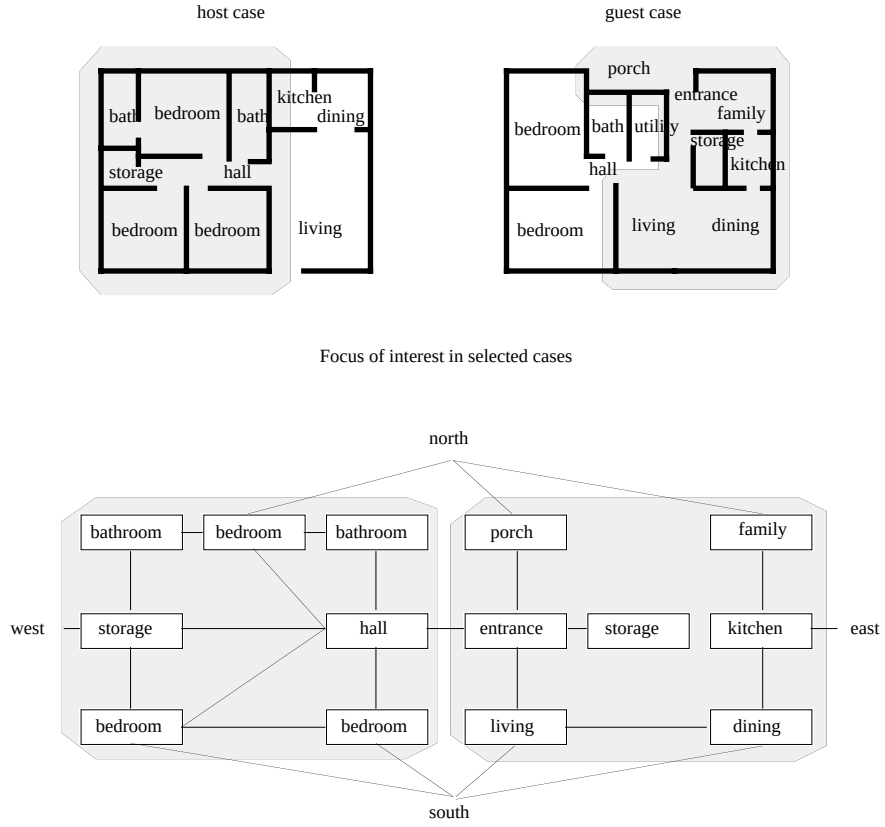

Fig. 6. Selected design cases and specifications for case combination

does not yet know what to do along the edges of case fragments. Next, the case fragments are inserted in the new problem context and new problem specific requirements are specified. This is facilitated by the use of geometry and topology-based representations, corresponding levels of detail, property sheets and direct manipulation. At this stage, it becomes necessary to introduce a new operator which allows a user to declare two or more items or elements from different case fragments as identity elements in some sense. This is typically done by the user dragging and dropping two elements on top of each other.

Once the new problem specific requirements are entered, a number of subsequent processes are invoked. The first process checks if all the design elements in case fragments have been *glued* to each other via some requirements. This is particularly important for circulation elements. In simple and straight forward case combination, the system treats the resulting set of constraints in a manner identical to case adaptation described previously.

On the other hand, it may happen that some design elements need to be treated with special attention. For example, there may be a circulation element in one case fragment that has not been explicitly related to anything in the other case fragment. In other words, the system needs to check and figure out what happens at the boundaries of case fragments. This phase of the system reasoning is still under development, further details about various strategies we are experimenting with are described in Section 3.4).

One additional notion that we introduce at this stage is to consider one of the contributing cases as the *host case* and the other as the *guest case*. This

Fig. 7. Results of case combination in 2D

distinction enables the system to treat the guest case as the one that has to be *adapted* to the host case, the solution of which together leads to the design solution of the new problem. This distinction has the following advantages: (i) to let the system execute the same process of case adaptation for the required number of cycles individually for each case fragment, and (ii) to preserve some architectural or structural properties, e.g. to carry over the structural system of the host case into resolving the structural system of the guest case. An example shown in Figure 7 illustrates the results obtained by the process of case combination for a simple design context.

3.4. NEAR TERM AGENDA

The current organisation of various modules in the system is illustrated in Figure 8. At present, most of these modules are implemented but exist separately and communicate with each other via a set of files. We are in the process of assembling these modules into an integrated environment with a top-level graphical interface. Various modules represent either the information that is entered by the user or that which is generated by the system in response to such information.

Interface: We have started work on building a graphical interface that supports all user specifications and also provides a visual and graphical feedback about whatever the system is processing at a given moment. Although a limited prototype interface has been developed (see Figure 3), we consider a more robust and complete interface implementation as one of our

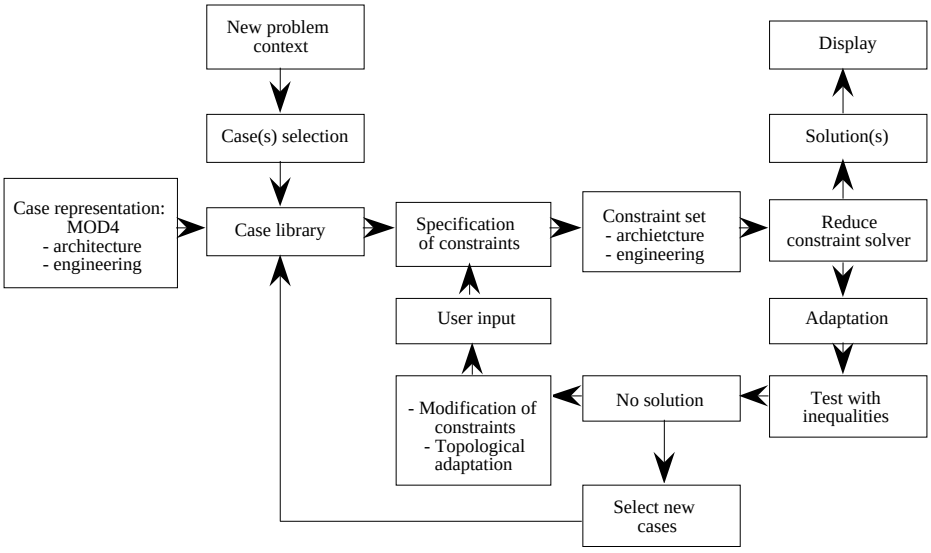

Fig. 8. System architecture

immediate priorities. Additionally, we are able to model and develop geometry and topology-based representations for cases but need more work to be able to support grammar-based graphical representations.

Case Combination: As shown in Figure 7, we are able to successfully apply case combination for a small design problem in 2D. We are currently working on three primary extensions of this process in 3D to obtain real volumetric combinations of cases. The first extension involves incorporation of some domain specific knowledge. For example, given parts of two cases to be combined in response to a new problem, the system first attempts to resolve them in two-dimensions, i.e. essentially placing all the spaces on one floor. If it fails, it stacks spaces such as vertical circulation and services on two or more different floors and moves associated spaces on relevant floors (Figure 9). This is followed by a process of case combination that runs separately for each floor but maintains certain constraints about vertically aligning some spaces and elements on separate floors. These ideas have been tested using two simple cases of residential designs that were combined for a new design problem, the results of which are shown in Figure 10.

The second extension requires the system to work with only partial but minimum recommended overlapping spatial units on two floors, e.g. in case

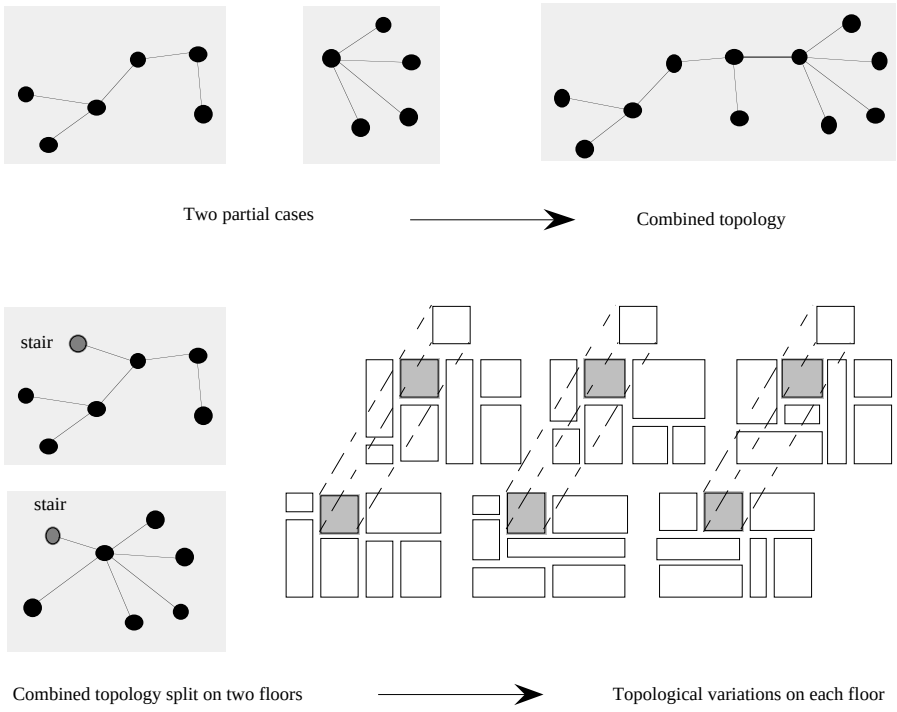

Fig. 9. Case combination in 3D

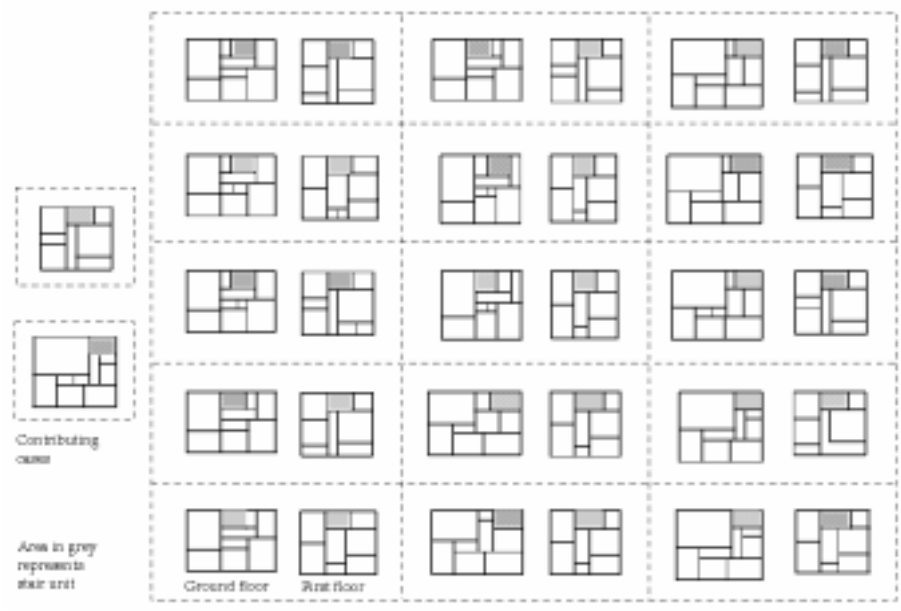

Fig. 10. Case combination in 3D

of service cores. A more complex example will also take into account an efficient design of structural system spanning across floors.

The third extension involves possible grouping of spaces into aggregate spatial units which will enable the system to apply the same case combination procedures but at increasing levels of detail. For example, the system aggregates all *public*, *semi-public* and *private* spaces in a building, and attempts case combination at a coarser level of detail before working on more detailed parameters of spaces.

Engineering Issues: Although we have presented our ideas primarily from the architectural standpoint, the system also incorporates structural design issues. As a result, whenever a case is inserted and parameterized for a new design problem, a corresponding set of structural engineering constraints are also generated in order to design and verify the resulting structural design of generated solutions. Occasionally, this leads to problems; for example, an adapted design may require placement of a structural column in the middle of a room for an otherwise well-adapted design solution. We are investigating the use of structural grids as one possible solution for resolving such conflicts as early in the adaptation process as possible.

Case Base: Once all the modules have been implemented and tested for small design problems, we intend to test these ideas further by working with large and more applied design situations. With this aim, we have built up a case base of building designs for architectural schools in various universities.

Our intention is to use these cases to generate new designs for architecture schools to evaluate how the system scales up for realistic, large scale design projects.

At present, we have selected four different cases (Figure 11) as candidates for inclusion in our case base. The design of architecture school at the University of Houston is an example of an extreme historical adaptation. The school at Rice University is an example of a new design that is adapted to blend in with its existing surroundings. The school at University of Arizona incorporates a number of design elements such as a *courtyard* that are recurring themes in architectural designs, whereas the school at Harvard University exemplifies a completely different design approach. Taken together, these four cases represent different design intentions and offer quite a diverse set of solutions for testing our ideas. These cases have been modeled from both the architectural and engineering viewpoints, i.e. the case base contains a geometric model enhanced with symbolic information for each of these buildings as normally referenced by architects as well as structural engineers.

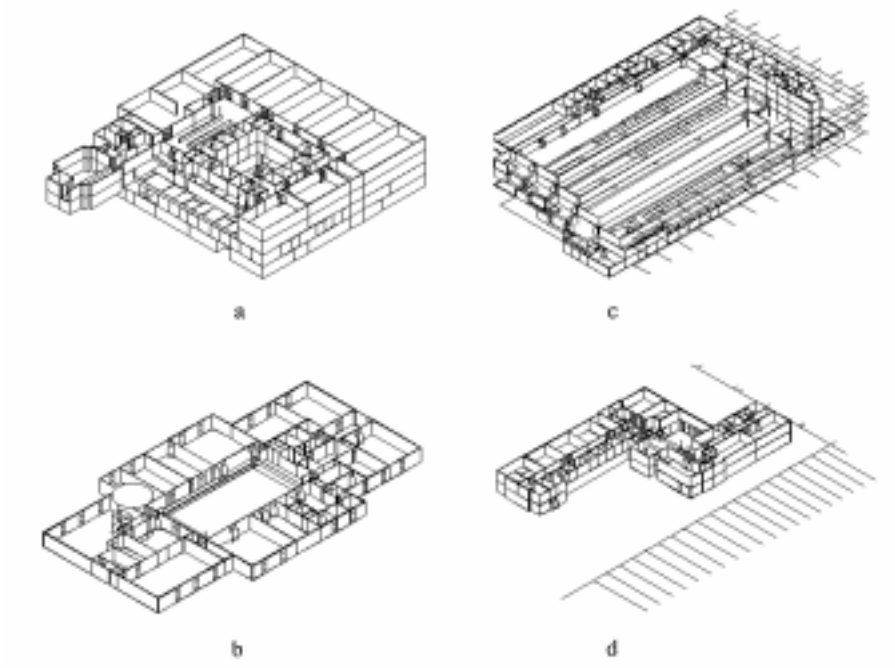

Fig. 11. Cases of architectural schools in the case base
(a) University of Arizona, Arizona (b) University of Houston, Houston
(c) Harvard University, Boston (d) Rice University, Houston

4. Discussion

In this section, we attempt to answer and raise some issues that were not sufficiently described in the preceding sections.

• *How big a case base is needed?* We have chosen to opt for a small case base instead of a large one but we have selected cases that are complex, large-scale and diverse. The main focus is on developing a representation mechanism that will support incorporation of new cases without substantial, if any, changes in the future. A larger case base will certainly offer more and perhaps better opportunities for generating innovative designs as a result of case combination. At the moment, we have been restricted in this regard partly as a result of realistic constraints of time. It is also partly a result of our view that we are pursuing this project in the domain of architectural design in which an all purpose design case base is difficult to develop a priori.

• *Why are cases not indexed?* Although most other CBR systems treat classification of features and indexing of cases as one of the central issues, we have chosen to view these issues in a different light, primarily providing one representation from which other useful *views* of information may be generated and used by individual designers. There are a number of reasons for this. First, our case base is too small to warrant any elaborate indexing mechanism. Second, classification of features in the architectural design is meaningful and useful only with respect to a specific purpose at hand. In fact, we think that individual designers *see* features in designs that are a function of their personal experiences and thus enable them to engage in creative associations that we do not fully understand. Third, even if we were able to classify and index a set of features, the case base will require complete updating and modification the moment a new feature is proposed in the future. Fourth, since we do not know in advance how various characteristics of design cases are to be used in either case adaptation or case combination, it is not very useful to classify or index any one characteristic. It is useful to know how cases are going to be changed before we decide on how they are going to classified and stored. For all these reasons, we do not see much usefulness in a priori and static classification and indexing case features.

We consider it essential that a case representation be used that allows individual designers to specify their own classification features, e.g. a designer may want to classify cases on the basis of *symmetrical plan* disposition or the use of *diagonal circulation* patterns. There should be a facile means for the designer to specify such a case feature and the system should be able to interpret, reorganise and present cases on that basis. Although we have attempted to address this need in our system by providing different representations and levels of detail that are either explicitly represented or computed on demand, we may benefit from looking at how

database systems allow users to create new logical views of tables by specifying *attribute-value* pairs on the fly. Ideally, such a facility should support a direct manipulation graphics interface.

• *Will the compiled knowledge be more efficient?* The system as developed does not know nor cares about how a given case was designed in the first place. For case adaptation and case combination, the system proceeds directly to the identification of discrepancies and then attempts to resolve those. This approach suffices for the kinds of design problems that we have worked with so far. But in the context of more complex design problems that we have in mind, we may have to resort to some generalised set of compiled knowledge in the form of rules. For example, if a particular case combination situation calls for locating spaces on different floors and more than one case has been used in the process, we will need a way to encode how to maintain circulation, services or structural continuity in the final design that results from combining a number of partial cases.

• *How are the posted constraints scheduled for resolution?* The process of dimensionality reduction is aimed at finding the subspace of parameters involved in discrepancies, i.e., functions that are violated after a case has been inserted in a new environment. This process does not prioritise or distinguish between various parameters, they are solved simultaneously. Thus, there is no need for a scheduler and there is no possibility of subsequent conflicts. The only time when this becomes an issue is when architectural and engineering constraints lead to mutually unsatisfactory design solutions perhaps as a result of topological modifications. At present, we do not have a clear strategy for handling such conflicts.

• *Is anything "learned" out of a given problem solving?* There is no record of how the system worked on a given problem and thus there is no facility for it to learn from its efforts. At a deeper level, we have so far chosen to work with first principles for case adaptation and case combination; as a result, we have not yet incorporated a learning mechanism.

• *What are the limits of this approach?* A comprehensive set of fundamental problems in case based design systems were presented in (Hua et al., 1992). By taking the approach reported here in our work, we have managed to bypass some of these problems. At the same time, some limitations of our approach are as follows. First, for every case adaptation and combination, dimensionality reduction has to be computed. Whether the identified set of parameters actually leads to a solution or not is not known until the whole process has run through. It would be useful to somehow combine dimensionality reduction with, on the one hand, a *look-ahead strategy* that can ignore unpromising parameterization and, on the other hand, *templates of solution fragments* that could be reused in future. Second, the topological adaptation process is still not fully resolved, leading to

occasional but expensive modification-generation-verification cycles. This may be improved to an extent by either adding knowledge of possible topological adaptations to cases or storing such knowledge in the form of separate rule bases. One other limitation of the approach is that it works with only rectangular spaces, a limitation that results from representation scheme adopted in the work.

• *Is this a model of designing?* Our work proposes some computational ideas that are capable of case adaptation and case combination but it would be inappropriate to suggest that it also provides a model (particularly a cognitive model) of designing. As suggested before (Section 2), many architectural designs are a result of creative adaptation and combination of previous design cases but a particular route taken by designers is continuously changing in response to a particular design context. In our work, we focus on those aspects of design development where computational strategies can be devised and may complement a designer's efforts rather than looking for complete automated solutions for all phases in design or simply replicating the traditional practises.

• *How is our approach different from other CBD projects?* One of the central features of many CBD systems, for example FABEL (Fabel, 1992), is classification and indexing of features, an issue to which we assign a low priority for the reasons stated previously. A system such as Archi-II (Domeshek and Kolodner, 1992) provides an extensive set of design descriptors to be used as indices to retrieve and present cases. We also share the position that information in the case base should be easily retrievable in a format appropriate to the domain under consideration. At the same time, our emphasis is on providing a few selected representations, using which the user can develop dynamically his or her wayfinding in the case base. Other systems focus on incremental resolution of constraints in order to adapt cases, for example, CADSYN (Zhang and Maher, 1993), whereas the dimensionality reduction procedure we use zooms in on the subspace of parameters thereby reducing iterative search cycles in constraint satisfaction. This also allows us to deal with variables that are continuous and also those that take on values within some intervals in a way that is distinct from other projects. Lastly, user interaction is considered an essential part of the process in our work, a feature that is perhaps not considered very significant in many other CBD systems. Finally, our work addresses design problems that are inherently spatial and thus requires us to work with symbolic information that represents three dimensional configurations of objects and the consequent experiential, behavioural and functional properties that derive from them. In this regard, our work appears to be closer to the issues raised in projects like ARCHI (Goel et al., 1991), in particular to how to apply CBR techniques in the domain of architectural design systems.

Acknowledgements

This research is funded by the Swiss National Science Foundation under the program NFP 23: Artificial Intelligence and Robotics. The authors would like to acknowledge contributions of the following project members: Shen-Guan Shih, Simon Bailey, Kefeng Hua, Laurent Bendel, Jean-Marc Ducret and Kim Jent.

References

Domeshek, E. A. and Kolodner, J.: 1992, A Case-Based Design Aid for Architecture, *in* J. Gero (ed), *Artificial Intelligence in Design '92*, Kluwer Academic Publishers, The Netherlands, pp. 497-516.

Fabel: 1992, *FABEL: Integration von modell- und fallbasierten Entwicklungsansaetzen fuer wissensbasierte Systeme*, Verbundvorhaben des BMFT (01IW104) unter der Leitunge der Gesellschaft fuer Mathematik und Datenverarbeitung, Bonn.

Goel, A., Kolodner, J., Pearce, M., Billington, R. and Zimring, C.: 1991, Towards a Case-Based Tool for Aiding Conceptual Design Problem Solving, *in Proceedings of Case-Based Reasoning Workshop*, Washington, pp. 109-120.

Hua, K., Smith, I., Faltings, B., Shih, S. and Schmitt, G.: 1992, Adaptation of Spatial Design Cases, *in* J. Gero (ed), *Artificial Intelligence in Design '92*, Kluwer Academic Publishers, The Netherlands, pp. 559-575.

Hua, K., Schmitt. G. and Faltings, B.: 1992, What Can Case-based Design do?, *in Proceedings of the AID'92 workshop on Case Based Design Systems*, Carnegie Mellon University, Pittsburgh, pp. 44-53.

Schmitt, G., Faltings, B. and Smith, I.: 1993, Case Based Spatial Design Reasoning, *in Proceedings of NRP 23 - Symposium on Artificial Intelligence and Robotics*, Zurich, pp. 11-16.

Shih, S. G. and Schmitt, G.: 1993, The use of post interpretation in grammar-based generative systems, *in* J. Gero and F. Sudweeks (eds), *Proceedings of IFIP Workshop on Formal Design Methods for CAD*, Key Centre of Design Computing, University of Sydney, Sydney, pp. 101-113.

Zhang, D. M. and Maher, M. L.: 1993, Using Case-Based Reasoning for the Synthesis of Structural Systems, *in Knowledge-based systems in civil engineering*, IABSE Colloquium, Beijing, 1993, pp. 143-152.