Zeitschrift: IABSE reports = Rapports AIPC = IVBH Berichte
Band: 68 (1993)

Artikel: Integrated case-based design systems
Autor: Hua, Kefeng / Smith, Ian / Faltings, Boi
DOI: https://doi.org/10.5169/seals-51875

Integrated Case-Based Design Systems

Systèmes d'étude intégrée rapportés au cas spécifique

Integrierte fallbezogene Entwurfssysteme

Kefeng HUA
Research Assistant, AI Lab.
Swiss Fed. Inst. Technol.
Lausanne, Switzerland

Kefent Hua received a Msc in computer science at the Beijing Inst. of Technology in 1985. Currently, he is a doctoral student at the Artificial Intelligence Laborator, EPFL.

Ian SMITH
Adj. Dir., AI Lab.
Swiss Fed. Inst. Technol.
Lausanne, Switzerland

Ian Smith received a Civil Eng. degree from the Univ. of Waterloo, Canada, and a PhD from Cambridge Univ., UK, in 1982. In 1988, he started the knowledge systems group at ICOM and in 1991, he joined the AI Lab.

Boi FALTINGS
Dir., AI Lab.
Swiss Fed. Inst. Technol.
Lausanne, Switzerland

Boi Faltings received a diploma from the ETH Zurich and aPhD from the Univ. of Illinois, both in Electrical Eng. Prof. Faltings founded the Artificial Intelligence Laboratory at EPFL in 1987.

SUMMARY

A design problem can be viewed from different abstractions. An architect e.g. sees a building as a collection of rooms with particular properties, while a civil engineer sees a structure of load-bearing elements. For design, it is important to combine these viewpoints into a single coherent object. Difficulties associated with combining viewpoints lead to the so called integration problem. Case-based design (CBD) is a recently developed knowledge-based technique for knowldege-based design systems. This paper describes how integration problems may be solved. Cases of previous design solutions provide the basics for the integration of several abstractions into a single object. When novel designs are created by adapting cases, integrity can be maintained through careful formulation of the adaption procedures. A prototype design system is described.

RÉSUMÉ

Il est possible d'envisager un projet sous différents aspects d'abstraction. Un architecte voit p.ex. un bâtiment en tant qu'assemblage de locaux ayant diverses propriétés, tandis qu'un ingénieur perçoit une structure porteuse constituée d'éléments constructifs. Pour la conception du projet il est important que les deux points de vue se rejoignent en un objet cohérent. C'est ce qu'on entend sous la notion de problèmes d'intégration. Les auteurs montrent comment la technique opératoire récemment développée pour les systèmes experts peut venir en aide dans les problèmes d'intégration. Les résultats d'études de cas spécifiques mis en mémoire mettent en évidence les possibilités de combinaison pour l'intégration de diverses abstractions. En développant de nouvelles études à partir de l'adaptation de cas précédents, il est alors possible de conserver l'intégrité par une formulation soignée des phases successives d'adaptation. L'article décrit un système prototype.

ZUSAMMENFASSUNG

Ein Entwurfsproblem kann aus unterschiedlichen Abstraktionsrichtungen angesehen werden. Ein Architekt z.B sieht ein Gebäude als Ansammlung von Räumen unterschiedlicher Eigenschaften, wo der Ingenieur ein Tragwerk aus diversen Bauteilen wahrnimmt. Für den Entwurf ist wichtig, beide Auffassungen zu einem kohärenten Objekt zu vereinen. Dies versteht man unter dem Begriff des Integrationsproblems. Es wird gezeigt, wie die jüngst für Expertensysteme entwickelte Methodik des fallbezogenen Entwerfens bei der Lösung des Integrationsproblems helfen kann. Gespeicherte Ergebnisse von Fallstudien zeigen Kombinationsmöglichkeiten unterschiedlicher Abstraktionen auf. Wenn neue Entwürfe aus der Anpassung früherer entwickelt werden, kann die Integrität durch sorgfältige Formulierung der Adaptationsschritte bewahrt werden. Ein Prototypsystem wird gezeigt.

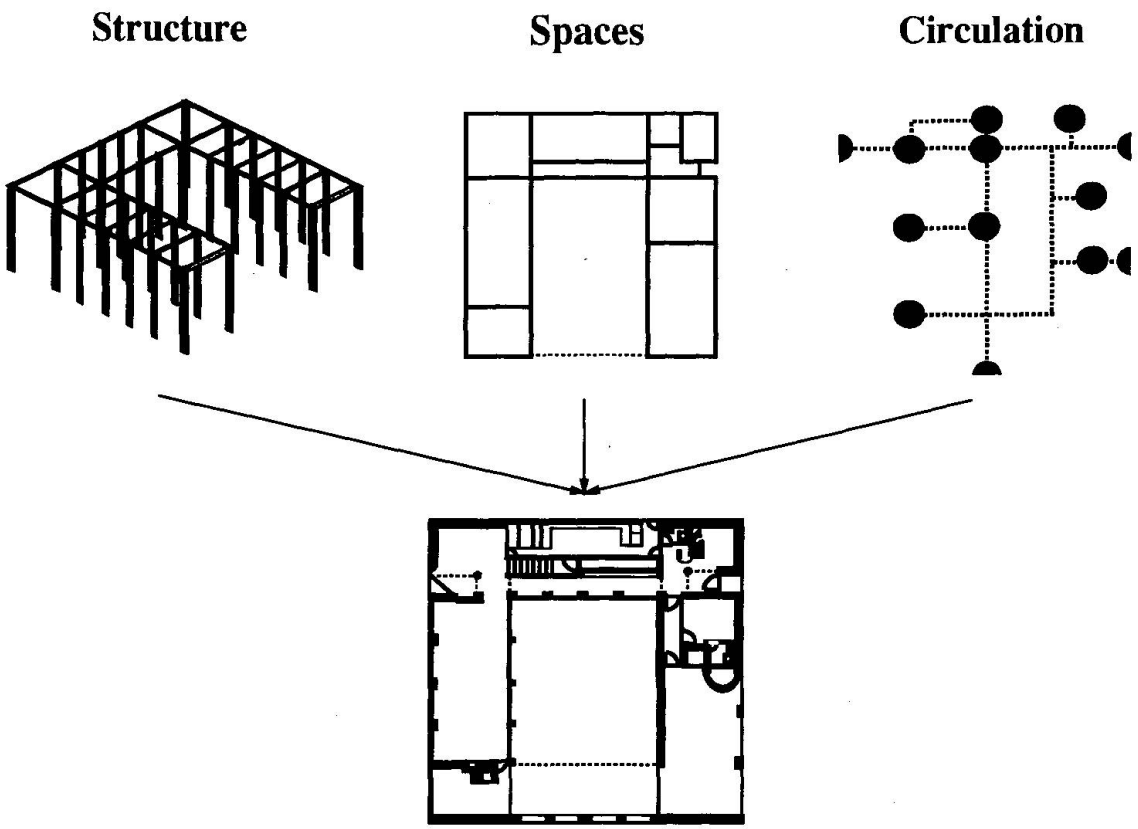


Figure 1: *A building represents an integration of many different abstractions, including structure, spaces and circulation pattern.*

1 The Integration Problem

Any physical artifact can be viewed according to many different abstractions. For example, a building can be:

- an ingenious civil engineering structure of beams and columns.
- a magnificent way of creating architectural spaces.
- a practical arrangement of functions for its occupants.

Designing a building is difficult because it has to *integrate* satisfactory solutions in each abstraction: the structure designed by the civil engineer, the spaces laid out by the architect, and the circulation pattern desired by the user are part of one single structure (Fig. 1).

Disagreements and misunderstandings between architects and civil engineers are recognized as sources of many problems in construction[1]. Producing and documenting designs on a CAD system, preferably an intelligent CAD system, help detect problems during the design phase through checking consistency between the designs produced by different people. Research efforts such as IBDE [9] have already proposed computer tools for integrating designs generated in different abstractions.

In IBDE, seven different modules correspond to different abstractions and communicate via a common data representation called a blackboard. Inconsistencies are detected by critics and cause reactivation of certain modules in order to eliminate the problem. Since corrections are constructed locally, this process may well cycle or even diverge, as illustrated by Figure 2: correcting the discrepancies by locally adjusting either P1 to satisfy C1 or P2 to fall onto C2 leads to a cycle which diverges from the solution. Only through simultaneous consideration of all abstractions can such problems be avoided.

[1]Disagreements involving occupants are probably even more frequent, but rarely communicated to the designers.

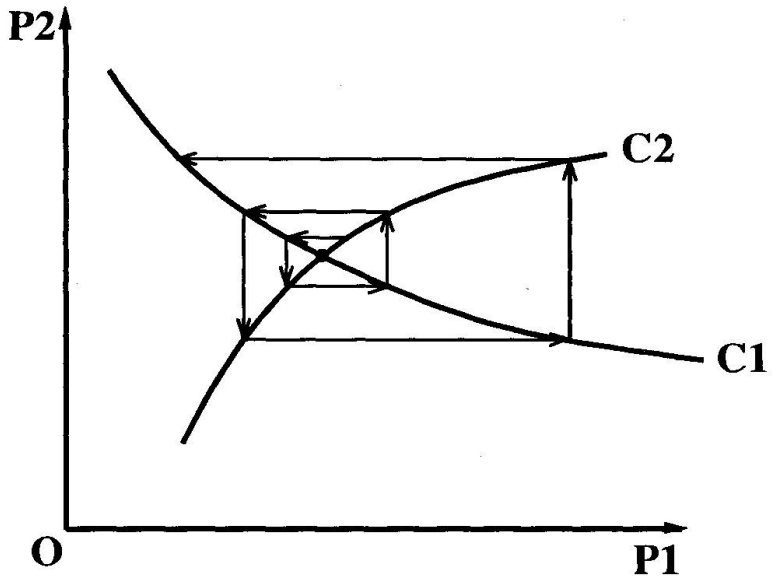


Figure 2: *P1/C1 and P2/C2 represent parameters and constraints in two different but interacting abstractions: the correct choice of the parameter is on the constraint curve. When discrepancies in each abstraction are corrected in isolation, the process may diverge as indicated by the arrows.*

Achieving integration in a classical knowledge-based system framework is in principle possible, but extremely difficult because there are few general principles which hold over all abstractions. Attempts to formulate knowledge in an integrated way exist. For example, Alexander [1] has produced a handbook which defines principles of good design that consider several abstractions simultaneously. A striking fact about his work is that the rules he defines are actually prescriptions for particular buildings in particular environments, with little generality. The lesson from this observation is:

> *Integrating design knowledge from many abstractions amounts to formulating particular cases of good design.*

This observation leads to the formulation of design knowledge as *prototypes* [3] which are generalized versions of particular structures. However, since prototypes still require tedious formulations of the generalizations that apply, design by reusing previous *cases* is of interest. In this paper, we show how this paradigm of *case-based reasoning* can be applied to solve the integration problem in building design.

2 Integration through Case Adaptation

Case-based reasoning From the beginning of AI, cases have been regarded as an important source of knowledge. For example, the checker player of Samuel[6] used a library of some 53000 cases of positions and demonstrated a performance of a better-than-average novice. Learning from examples is a fundamental strategy of knowledge acquisition, and could be applied to design cases. The main difference between learning from examples and case-based reasoning is that instead of generalizing cases during knowledge acquisition, they are generalized with respect to a specific problem during the problem solving process itself.

Case-based reasoning originates from psychological models of human memory structure [8]. A case-based problem solver consists of two processes: *indexing* to find a suitable precedent, and *adaptation* to use it in the new problem context. Although the indexing problem has been studied in the literature, known schemes rely on symbolic features which are hard to define in design. The adaptation problem has only been addressed in very simple domains. For case-based *design*, adaptation is essential; no

two design problems are ever identical. Since indexing can be carried out by user interaction, we have focussed our research on the adaptation of cases to new problems.

Design cases Design requires knowledge in order to synthesize structures. For building design problems of realistic size, formulating such synthesis knowledge is very tedious, since conflicting goals lead to many tradeoffs. This knowledge is more easily accessible in the form of cases of existing buildings, and each case incorporates a large amount of synthesis knowledge.

A case-based design system can be characterized by its dependence on cases as an exclusive knowledge source. We define a *shallow* case as a model of an existing building without any further information about how it was obtained. In contrast, a *deep* case is augmented by a trace of the process which devised the design. Although additional information in a deep case can be used to guide indexing and adaptation processes, its interpretation would require a design knowledge base which is sufficiently complete to generate the design. Since the main point of using cases for design is to avoid having to formulate this knowledge, we attempt to limit our research to cases which are as shallow as possible in order to test how far this approach is applicable.

What is a case? A shallow case defines an actual artifact, represented for example as a CAD model of the actual building. In our implementation, we use AutoCAD as a tool for representing and rendering this information.

Buildings are examples of integrating functions[2] according to different abstractions. Our prototype considers spaces, circulation and structures as examples. These functions are modeled by a symbolic vocabulary appropriate to the corresponding abstraction, and mapped to *constraints* formulated on the common CAD model. The CAD model thus serves as a basis for integrating different abstractions. A case defines a set of "good" ways of achieving functions in different abstractions, and a way to integrate them into a single building.

Case adaptation Applying a case to a new problem requires changing the structure while maintaining the integration of the abstractions that has been achieved in the case. The fundamental assumption underlying case-based design is that *adaptation is easier than generation.* There are three reasons why this assumption is reasonable:

- It is often impossible to formulate explicitly the knowledge required to generate designs involving complex tradeoffs.
- If the case is sufficiently close to a solution to the new problem, only few changes are required.
- Many details of the case, such as the type of windows, can be carried over to the new solution.

Case-based design is a successful paradigm for solving the integration problem only as long as these reasons are valid. For example, dimensioning of simple elements and other tasks found in routine design activities may not contain requirements which are necessary for effective implementation of case-based design strategies.

In the methodology we developed, adaptation consists of the following processes in order to ensure that such advantages are exploited:

1. *Insertion* of the case into the new environment.
2. Determination of the *discrepancies*: functions which are no longer satisfied due to the new specifications.
3. *Parameterization* of the relevant parts of the case in order to eliminate the discrepancies. Parameterization is understood in a general sense, covering both dimensional and topological aspects.

[2]We use the term *function* to denote any feature; structural stability is also a function.

4. *Modification* of the case into a new solution.

Insertion is the process of determining a match between the original environment of the case and the environment of the new problem. Finding the match is often a difficult and ambiguous problem, and is achieved with interactive help from the user. **Discrepancies** occur when differences between the specifications of the case and the new problem cause certain functions to disappear. An example of a discrepancy is that the building exceeds restrictions imposed by the new site, or that it provides too little floor space.

The parts of the case involved in discrepancies are those which need modification in the new solution. **Parameterization** of the case collects those parameters which are part of the discrepancies or which are related to them by constraints. Topological adaptation might be achieved by representing the topology as a grammatical structure when elements can be exchanged. Dimensional **modification** of a case consists of finding parameter values which give a feasible solution for the new environment. Topological modification is the addition, suppression or rearrangement of parts, and is always followed by renewed parameterization of the new structure. Modification is the process where the integration between the different abstractions must be maintained.

Maintaining integration through dimensionality reduction In dimensional modification, the different abstractions are represented as parameters and constraints between them. Constraints can be *definitions*, such as

floor-space = width * length

or *restrictions*, such as

width > 1.5m

All constraints can be mapped to parameters defined in the CAD model. Maintaining the integration during adaptation means that any modification should leave all constraints which are currently achieved intact; the modification must remain within the *subspace* of the parameters defined by the constraints. This subspace can be explicitly constructed and parameterized using a *dimensionality reduction* [7] process illustrated in Figure 3.

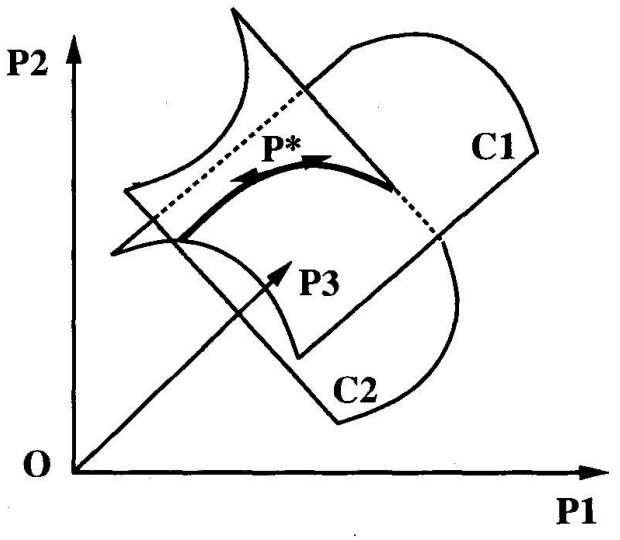
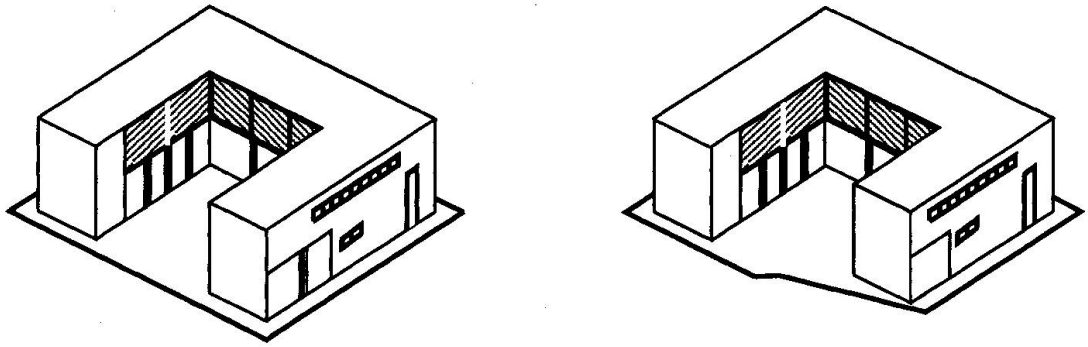


Figure 3: *P1/C1, P2/C2 and P3 represent parameters and constraints of three different abstractions. By defining a new parameter, P*, which maps to positions on the intersection curve of the two constraints C1 and C2, solutions can be found without unstable iterations through selecting a suitable value for P*.*



Case **Adaptation of the case**

Figure 4: *Example of case adaptation*

Modifications using the reduced set of parameters can never create any contradictions between different abstractions, and thus cycling or diverging behaviors common to blackboard systems (illustrated in Figure 2) do not occur.

Dimensionality reduction only applies to equalities. Among inequalities, we can distinguish two types: *critical* inequalities which are limitations exploited to the maximum and just satisfied in the case, and *non-critical* ones which are satisfied by a large margin. If the case is sufficiently close to the new solution, critical-constraint sets can be assumed to remain the same in spite of the adaptation. Thus, critical inequalities can be replaced by equalities to which dimensionality reduction applies. Non-critical inequalities are constraints on the parameter values to be chosen for the modified instantiation and are handled by constraint propagation mechanism.

Topological changes For topological changes, we have not yet succeeded in defining an analog to dimensionality reduction; in fact, such an analog may not exist. Thus, we cannot ensure that integration is maintained throughout a topological modification. However, case adaptation still offers advantages over generation; if the case is sufficiently close to a feasible solution, the number of topological changes that are required, and may destroy the integration, is much smaller than what would be entailed by generating the building from scratch.

Currently, topological adaptation proceeds in the same way as other design systems, e.g. IBDE [9]. Different knowledge sources act on different abstractions. The user re-establishes constraints in the new topology through constraint posting in order to follow this step with a new dimensional adaptation, thereby ensuring that the new topology meets dimensional requirements. The solution is subsequently checked for consistency with non-critical constraints originating from all abstractions. If this check fails, the current proposal for topological adaptation is rejected and the system returns to the relevant knowledge sources for another proposal.

3 CADRE, A Prototype Design System

In order to explore the adaptation of cases in design, we have implemented a CAse-based building design system through Dimensionality REduction(CADRE) [4, 5]. One example treated by CADRE is shown in Figure 4. It is a U-shaped building (the Felder House in Lugano, Switzerland, [2]) adapted to a slightly different site. CADRE modified both the dimensions and the topology of the case in order to obtain a solution that preserves the functionalities and tradeoffs in the case.

Computationally, the processes in CADRE can be divided into two layers: a symbolic layer and a numerical layer. They correspond to the topological and dimensional models of the case. CADRE focuses on case adaptation, and leaves case selection to the user. The adaptation is conducted with the following procedure:

1. Evaluation of the existing case in the original and new environments in order to find discrepancies. Insertion of the case into the new design context so that a maximum coincidence is achieved, subject to constraints posted by the user.

2. If there are dimensional discrepancies, identify the violated constraints and the parameters which are involved in them. Complete the set of applicable parameters and constraints with all those which are related to the original ones through links in the constraint network. This defines the complete *base set* of parameters and constraints related to the discrepancies.

3. Apply *dimensionality reduction* to the base set of parameters and constraints to define an adaptation parameterization which is guaranteed to avoid conflicts.

4. Modify the dimensions using the parameters resulting from dimensionality reduction. The user controls the process by asserting additional constraints or manually identifying suitable values.

5. Check the validity of the adaptation by verifying inequality constraints in the base set that were not critical and thus not treated by the dimensionality reduction.

6. If there is no solution at the dimensional level for the new design problem, trigger topological transformation rules which relax constraints in the related constraint set. If there is a transformation which preserves design features of the case, go back to step 1, otherwise the case is not suitable.

The next section gives an example of how this procedure has been implemented in CADRE. Tests on several real examples, along with discussions with practising engineers and architects lead us to believe that the procedure described above is complementary to their activities.

4 Example - Adaptation of a Building

In order to illustrate CADRE, a part of the Computer Science building group, the INR building, at the Swiss Federal Institute of Technology will be used, see Fig. 5. This building was designed to be a multi-purpose research building so that it can accommodate any technical research activity. The architect and the engineer re-used a design employed for a similar building on the same site. Construction was completed in 1992.

-1z Coordination of the adapted case with respect to all abstractions was not entirely successful. More specifically, some rooms which were laboratories in the original design were changed into

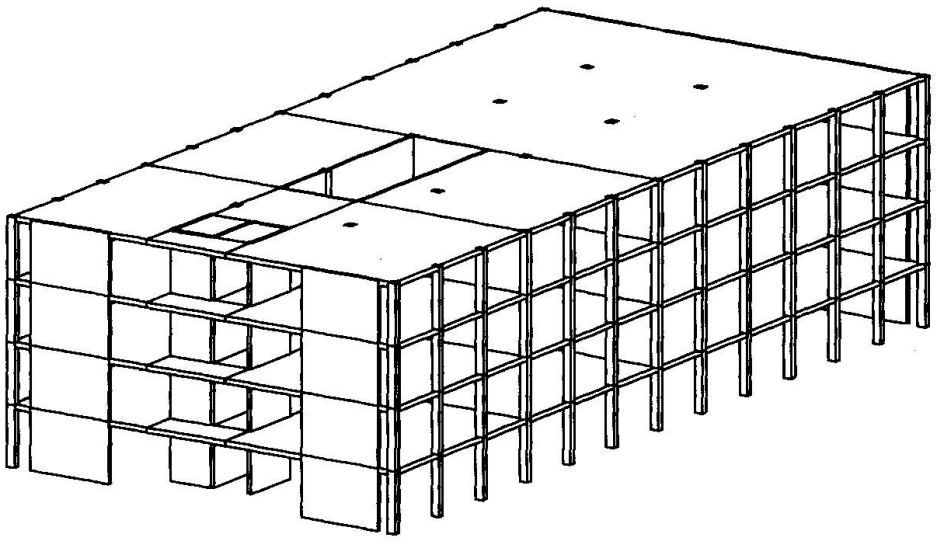


Figure 5: *3D-view of the structure of the INR building.*

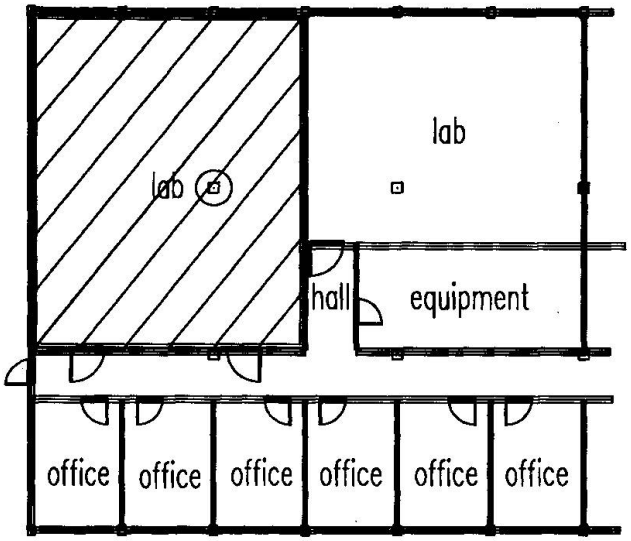


Figure 6: *A partial second floor plan of an earlier building. The hatched area indicates the laboratory space which is used as a classroom in the INR building, recently finished. The blackboard is positioned along the wall on the left hand side of the building.*

classrooms for the INR building. Such change in use modified functional requirements for these rooms. Column positions which were acceptable in the free space of laboratories became design faults in the new case when these rooms became classrooms – students require unobstructed views of blackboards. This is not an unusual situation; such functionality defects are common in nearly every building. They represent most clearly the effects of poor communication between engineers and architects.

For this example, it is assumed that the case as stored is the as-built structure conforming to the original design (tolerating columns in laboratories). Part of the second floor configuration is shown in Fig. 6. Column positions are indicated as small squares. Load carrying frames are oriented vertically and are spaced at every second office position, corresponding to every second window bay. Thus, the exterior upper and lower sides of the building contain twice as many columns as a parallel line of interior columns. Interior columns are positioned along one side of the hall and within laboratory spaces.

The room in the upper left position was originally designed to be a laboratory and is stored as such in the case base. The new environment where this case must be inserted is the same as the original except that the functions of some rooms have been changed from laboratory use to classroom use. Therefore, the discrepancy detected upon insertion of the case into the new environment is the requirement that columns are no longer allowed in rooms susceptible to be changed into classrooms.

The program proceeds by selecting all constraints (stored in the base parameterization with the case) that are related to this discrepancy. These constraints are then combined with the new constraint requiring columns to be placed in walls in order to begin dimensionality reduction. This process is a problem-specific parameterization performed at run time. Once complete, dimensional adaptation is attempted in order to find a solution which does not involve changes in overall room layout. However, a restriction on the minimum size of the classroom causes this step to fail. Without such a constraint, a solution involving a classroom equal to the frame spacing (every two offices) would have been proposed. Note that such a proposal would have already been consistent with constraints in *both* structural and architectural abstractions. This is the advantage of dimensionality reduction using constraints originating from different abstractions; divergent looping is avoided.

The next step involves triggering topological transformations in order to relax the constraints included in the dimensionality reduction. A structural topological adaptation module, driven by rules governing acceptable topological changes, proposes a solution involving a load carry frame every three window bays rather than every two in the original case. This new configuration is placed back into the base parameterisation module and a new dimensionality reduction is determined. This time,

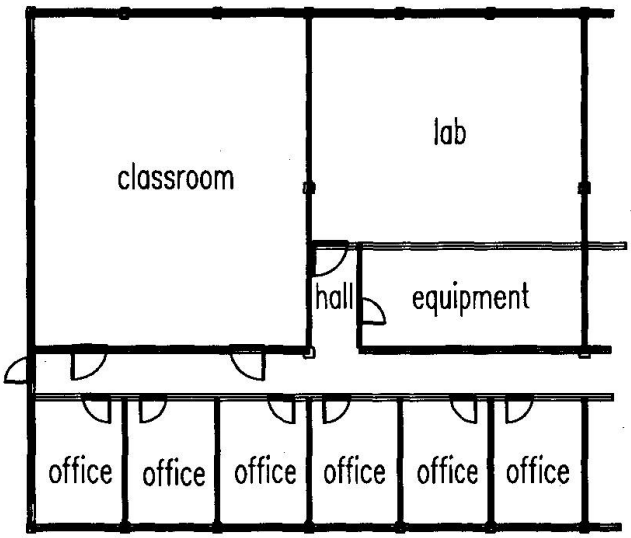


Figure 7: *The new configuration for the second floor of the INR building after structural topological adaptation and after an additional cycle of dimensionality reduction and dimensional adaptation. Constraints related to both architectural and structural abstractions are satisfied.*

dimensional adaptation succeeds; dimensions for elements such as floor slabs and column sizes are adapted to the new span length between frames. The new configuration is shown on Fig. 7.

This adaptation creates a new discrepancy on the first floor, where a column is now in the center of a room. Unfortunately, there is not enough space in this paper to complete this example with the figures necessary to illustrate subsequent steps. Briefly, architectural topological adaptation is triggered on the first floor to solve this discrepancy and a new parameterization and dimensionality reduction adjusts dimensions and ensures that all constraints are satisfied.

CADRE terminated by proposing a workable alternative for all floors without placing columns in rooms susceptible to becoming classrooms. We believe that had there been a tool similar to CADRE available to the engineers and architects during the design process, this building would have been built according to the configuration proposed in Fig. 7.

5 Conclusions

We have argued that case-based reasoning offers assistance for the problem of integrating different abstractions in design. Our prototype system, CADRE, illustrates the usefulness of the approach for practically interesting designs. The paradigm of case-based design fits very well with the observation that human designers like to work by reusing cases of previous designs. The considerations we have presented in this paper may be an explanation for *why* this is the case: integration of abstractions may be the main reason for designers to reuse previous cases. Adaptation of single cases is suitable for *routine* design. For innovation, we have to address the *combination* of cases; this is the topic of our current research.

Acknowledgements

This work is a result of collaborative research with CAAD(Computer-Aided Architectural Design), ETH Zürich, and ICOM(Steel Structures), EPF Lausanne. Discussions and collaboration with Professor Gerhard Schmitt (CAAD) have been most valuable. We would also like to thank the collaborators Shen-Guan Shih and Simon Bailey for their work on implementation of the ideas described herein, and to whom the credit for many of the details of the work is due. We also thank the Swiss National Science Foundation for funding this research as part of the National Research Program 23 on Artificial Intelligence and Robotics.

References

[1] **ALEXANDER, C.** "Notes on the Synthesis of Form" Harvard University Press, Cambridge, Mass, 1964

[2] **MARIO CAMPI - FRANCO PESSINA** Architects, Rizzoli International Publications, New York, 1987

[3] **BALACHANDRAN, M., GERO, J.** "Role of Prototypes in Integrated Expert Systems and CAD Systems" International Conference on Artificial Intelligence in Engineering, Boston, 1990

[4] **FALTINGS, B.** "Case-Based Representation of Architectural Design Knowledge" Computational Intelligence **2**, North-Holland, 1991

[5] **HUA, K., SMITH, I., FALTINGS, B., SHI, S. and SCHMITT, G.** "Adaptation of Spatial Design Cases" Second International Conference on Artificial Intelligence in Design CMU, Pittsburgh, USA, June 1992, pp559-575

[6] **SAMUEL, A.L.** "Studies in Machine Learning Using the Game of Checkers" IBM J. Research and Development 3:210-229

[7] **SAUND, E.** "Configurations of Shape Primitives Specified by Dimensionality-Reduction Through Energy Minimization" IEEE Spring Symposium on Physical and Biological Approaches to Computational Vision, Stanford, March 1988

[8] **SCHANK, R.** "Reminding and Memory" Chapter 2 in: *Dynamic Memory - A Theory of Reminding and Learning in Computers and People*, Cambridge University Press, 1982

[9] **SCHMITT, G.** "IBDE, VIKA, ARCHPLAN: Architectures for Design Knowledge Representation, Acquisition and Application" in H. Yoshikawa, T. Holden (Eds.): Intelligent CAD II, North Holland, 1990