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

Rubrik: Session 5: Validation, integration, and technical tools

SESSION 5

VALIDATION , INTEGRATION, AND TECHNICAL TOOLS

Integration of Building CAD/CAE Systems

Intégration des systèmes de CAO/IAO dans le bâtiment

Integration von CAD/CAE-Systemen im Hochbau

A. RECUERO
Dr. Eng.
I.C.C.E.T.
Madrid, Spain

Alfonso Recuero, born 1944, has been working for the past 20 years in computer methods for structural analysis. He leads a research project on the CAD Systems in construction. Alfonso Recuero became blind in 1974.

O. RIO
Dr. Eng.
I.C.C.E.T.
Madrid,Spain

Olga Rio, born 1955, obtained her doctorate in Civil Engineering at the University of Madrid. She has been working in structural analysis at the Torroja Institute for the past 10 years.

J.P. GUTIERREZ
Dr. Eng.
I.C.C.E.T.
Madrid, Spain

J.P. Gutiérrez, born 1950, has been working for the past 16 years in computer methods for structural analysis. He leads a research project on the Residual Life of Structures.

SUMMARY

CAD/CAE systems are an essential tool in the building design process. Although offering powerful mathematical models and sophisticated graphical capabilities to the designer they are not actually improving the design process model itself and the diffusion on expertise throughout the industry. It is necessary to create an integration environment developed around an open and expandable data base able to handle the building design conceptual objects and its data structure and organization. The objectives of this research project are presented in this paper.

RÉSUMÉ

Les systèmes de conception et ingénierie assistées par ordinateur constituent un outil devenu indispensable dans l'établissement des projets de bâtiments. Bien qu'ils fournissent au concepteur un modèle mathématique superpuissant et des possibilités graphiques des plus sophistiquées, ils ne sont pas encore à même d'améliorer le processus d'étude et la propagation des connaissances dans la branche concernée. Afin de développer une banque de données ouverte et extensible, il est nécessaire de créer un environnement d'intégration qui puisse permettre de faire fonctionner les objets conceptionnels du bâtiment, tout comme leur structure et organisation des données. Cet article présente un projet actuel de recherche dans cette direction, dont les objectifs sont largement explicités.

ZUSAMMENFASSUNG

CAD/CAE-Systeme sind ein unverzichtbares Werkzeug im Gebäudeentwurf. Obwohl sie dem Architekten leistungsstarke mathematische Modelle und ausgefeilte graphische Möglichkeiten an die Hand geben, verbessern sie nicht wirklich den Entwurfsprozess und die Wissensverbreitung in der Branche. Es ist nötig, um eine offene und ausbaufähige Datenbank eine Integrationsumgebung zu schaffen, die die konzeptionellen Gebäudeentwurfsobjekte, ihre Datenstruktur und -organisation handzuhaben gestattet. Der Beitrag behandelt ein Forschungsprojekt in dieser Richtung, dessen Ziele erläutert werden.

1 INTRODUCTION

Construction projects are developed in three consecutive parts: design, planning and construction. The first phase, that of design, represents only 10% of the total project cost but it decides on the remaining 90%. It is therefore critical to improve the performance on the construction design cycle, not only to reduce its direct costs but also, and more important, to enhance the quality of the decisions made in this phase which affect the construction itself and its life-time performance.

Computer aided design (CAD) and computer aided engineering (CAE). Although offering powerful mathematical models and sophisticated graphical capabilities to the designer they are not actually improving the design process model itself and the diffusion of expertise throughout the industry. They are a reflection of the fragmentation of many building design disciplines and their specialization. An architectural system does not communicate with a structural system and vice versa, which means that at least 50% of the information and other work has to be done again. This results in poor decisions and down grade performance of the final construction.

It is necessary to create an integration environment developed around an open and expandable database, able to handle the building design conceptual objects and its data structure and organization. The current research is directed to this integration, stating that the low and declining productivity of construction industry compared with that of manufacturing industry is due to the lack of coordination among the different experts who serve the building design process. This project will cover buildings as well as prestressed bridge construction, although the present paper will only deal with building construction. The aim is to create an open integration environment for computer aided building design consisting of a well documented building design process model; a reliable conceptual model of the building entity clearly defining the building objects hierarchically and the corresponding data structures; a database management system reflecting the conceptual models and dealing with objects rather than data; and common principles for user interface which meet the specific needs of building design.

This should improve the quality of the decision-making of the building designer, the overall productivity of the building design process, the management of the building construction, the cost performance ratio, the cost of the actual construction and the building maintenance and repair cycles.

2 THE BUILDING DESIGN (BD) PROCESS

Construction of Buildings has attracted the attention of mankind since the beginning of civilization. Regardless of the status of science and technology throughout history, man constructed various buildings, some of which are still standing, as witness to the wisdom, the will and the power of the ancients.

In every case and without any exception, three main phases were applied in the construction of a building. There was always, first, a Design Phase where the needs, the form and the materials of the construction were defined. Then, a construction Planning Phase had to be accomplished, in order to manage the available resources, the design requirements and the time limits. Finally, there followed the actual Construction Phase to materialize the Building according to the design and the resources. The only difference, throughout the ages, has been the supporting technology.

The Egyptian Pyramids were designed by inspired, "empty-handed" engineers and were constructed by people, rock and some primitive tools and mechanics, taking many years, for each one, to be completed. Today's New York Skyscrapers are designed and scheduled by computerized engineers and constructed by people using sophisticated materials and Robots, taking only a few months to build.

There is a close interaction between the three phases of the construction process. The Construction Planning may alter the designer`s decisions and always imposes minor or major modifications to the initial plan. The construction of the Building provides the designer with feedback information which may cause him to alter minor design details or update the initial design with last-minute, on-site changes. The latter is extremely important for the maintenance cycle and the repair program of the Building, and it would be a great benefit if the documentation of the final construction could be stored in detail, integrated, updated, easily accessible and retrievable.

The BD process can be divided into 10 main, distinct phases, each one of which is served by completely different traditions, sciences, methodologies, disciplines, expertise and practices. A listing of the BD activities, in a reasonable sequence can be:

1. Building Specifications
2. Landscape Architecture
3. Urban/Environmental Design
4. Architectural Design
5. Civil Engineering
   . Structural Design
   . Structural Analysis
6. Energy Engineering
   . Insulation
   . Heating
   . Air Conditioning
7. Facility Engineering
   . Electrical
   . Plumbing
   . Piping
   . Under-Piping
8. Sound Engineering
9. Interior space design / Decoration
10. Material selection and Bill of Materials

The BD process is basically sequential but also includes a lot of iterations and interactions between its various activities. Being sequential means that an activity has to be completed first, before the next activity can be developed. Although iterations may occur, the model has to follow the sequential path and run, every time, from top to bottom, in order to pass the changes made to the next activity.

3 PROBLEMS OF THE BD PROCESS

The previously mentioned list of BD activities could be shorter or longer, depending on the view and the emphasis placed by the given approach to the subject. But, short or long, the items are sequentially developed and successively processed. This results in a set of problems.

One serious problem arising from this sequential nature of the BD activities

is that the decisions taken in one activity impose serious limitations on the next, in such a way, that there is little or no room for decision- making in the last stages of the BD process.

Usually supported by independent computer systems, this sequential hierarchy of activities faces serious limitations from the conflicting decisions and/or from the non-compatibility of the supporting systems.

A second problem is lack of sensitivity. In everyday practice, the BD process is more or less a one-way process. The iterations and the interactions may occur only when facing dead-ends.

It is common to see an Architect who disregards some critical client's requirements in favour of the aesthetic, a Civil Engineer who ignores the functionality of the interior space in favour of the stability of the buildings, or a Facility Engineer who constrained limited by the previous two, "reconstructs" the building in order to find the "critical path" for the piping.

The cost, the time and the effort required to review any previous design step, in order to improve the performance of the next steps, is rarely undertaken. On the other hand, this "one-way" of the process is strongly fostered by the wide distribution and fragmentation of the expertise throughout the Building Design industry. The cost of coordinating individual experts who are trying to maximize their own performance, is always against the optimization of the final result. And all these without referring to the degree of the competition between the experts, that reduces even more the quality of the final design.

Even when supported by computer systems, a true sensitivity analysis (recurring "what if" capability) is rarely found in DB systems. Computers may improve the decision review capabilities of a designer, but in reflecting the fragmentation of the expertise, they too are specialized as well. The systems, being dedicated, cannot communicate, thus causing a lot of re-creation and re-processing of common information.

Being sequential, the BD process deals with common extended information which, in most cases, is recreated in each step. It is estimated that an Architect needs at least 25% of the Civil Engineer's information, while the Civil Engineer uses 50% of the Architectural information for his/her design, information that already lies in the blueprints of the Architect and the Structural Designer. Another problem is that the wide experience gained from the numerous construction projects by the many experts of the industry is presently stored in the personal memory of the experts who may comunicate and deliver a part of it to newcomer experts, in the form of on the job or on-site advices or University courses. This is not the best way of saving information.

A lot of expertise "dies" because there is no way for it to be stored permanently. Finding a way of storing this expertise will be a blessing.

4 STATE-OF-THE-ART

Architectural, Civil Engineering and Construction (AEC) computer applications, are among the oldest and represent a major application area because of the sheer size of the projects involved. (Approx. 478 billion ECUs were invested in construction during 1990, in Europe).

As previously stated, the Building Design process is performed today by a

multi-disciplinary and rather sequential set of highly fragmented and specialized expert activities. The existing computer applications imitate, in general, the same fragmented scheme, satisfying particular expert needs but ignoring the high degree of work duplication, cumulatively generated along the sequence of the successive Building Design activities.

The CAD (D for Drafting , not for Design) Systems available, being general purpose drafting tools, though impressive with their capabilities in visual outputs (3D shading models, plots,etc.), cannot deal with the features of the building objects (walls, columns, pipes, etc), while the mathematical modelling and simulation applications (FEM, BEM, etc.) are too specialized and poor in drawing capabilities to provide the final construction blueprints.

The existing integration of AEC systems is mainly based on the data exchanges between systems, used by the different Building Design actors, using either standard neutral file-formats or conversion schemes translating data from one system to another. Still, this technology deals only with graphics which, though essential, cover only a small part of the Building object.

An acceptable degree of integration of Building Design disciplines is provided only by specialized, private, closed architecture and rather expensive integrated environments, on Minis or on Workstations (ex. INTERGRAPH AEC family of applications on VAX and PC based MicroStation).

Considering that the low and declining productivity of the construction industry, compared to that of Manufacturing industry, is due to the lack of coordination (read : integration) between the "islands of expertise" that serve the Building Design process, current rese. rch is re-directed towards open integration of AEC applications.

There are already some initiatives that promote the idea of open integration in the AEC industry worth mention, such as CIFE (Center for Integrated Facility Engineering) of Stanford University (USA), CIB Working Commission 78 (Conseil International du Batiment), the RATAS project from VTT (Finland), the CAD/CAM Data Exchange Technology Centre, Leeds (UK), the AEC Systems series of conferences, and the work done at the Instituto Eduardo Torroja, Madrid (Spain), to mention a few.

5 ADVANTAGES OF INTEGRATION

Open Integration is defined as the act or process of making different complementary systems behave, though modular, as one.

Integration is not the merging of systems but rather a cooperation and coordination of systems directed towards a synergetic result.

Integration may overcome most of the previously mentioned problems of the BD process improving not only the performance of the process itself but also the quality of the final construction with considerable tangible and non-tangible benefits.

Through Integration, and without violating the basic flow of the BD process, the decisions taken in any stage could be evaluated in a very short time under multiple-criteria incorporated, at the beginning of the process, by the different actors. Integration then will play the role of the "manager" , who is now absent among the many "islands of expertise" of a BD project.

The cost of revising the decision will be more cost effective, compared to the benefits of improving the cost and the quality of the Building. The iterations and the interactions will be feasible, as the systems communicate automatically, transferring the required data from one system to another, in a very short time. The "what if" analysis and the design alternatives may become every day practice, while the experts will recognize their own potentials and drawbacks on the "mirror" of the display, looking at or reading the combined results of their decisions.

All Building data would be kept in the same format and in some cases in the same mass storage. The information would be passed from one expert to another through a diskette, if they were independently, or through an access call, if the experts are linked together on a Network or a Multi-User system. The time saved may then be invested in exploring better construction alternatives and in more-timely responses to deadlines and bids.

Integration alone cannot ensure intelligent storage of information, but it can lead the way to achieving this. Integrated environments can and should be linked with AI and Expert systems to analyze the information gathered and transform it into rules, statistic and case studies, for the benefit of the younger experts.

There may be many alternative solutions applicable to the BD process productivity problems.

We think that integration should be promoted through their research effort rather than Segmentation (Specialization), as the first appears to be a more justifiable and challenging solution for the BD process problems. In our opinion, integration respects and absorbs, from each individual system, its unique view, while it coordinates and directs all the outcome to one final, multi-aspect concept or result. Through systems integration the designer could run up and down, among any group of activities, even from top to bottom and bottom-up, as much as he/she feels it is necessary in order to spot a more efficient and less conflicting resolution before any final, no-way-back decision. With integration "What if ..." would then be feasible, while "What's best ..." would sound a lower-risk adventure.

6 INTEGRATION STEPS

Training of a company's staff in multi-disciplinary systems is not at all a productive investment, due to the increased turnover, especially in expert jobs. Trusting human ability not to make errors in transferring information from one system to another often proves to be an unforgivable mistake. On the other hand, independent experts are always too busy to learn to use more than one system.

A "first degree of integration" is based upon data exchange techniques that may export data from one application and input them to another. This method is more or less the common practice of the software industry.

By-passing the particularities of the different data structures of each application, these data-exchange files transfer limited and essential information only. IGES, STEP and DXF-files are some of the methodologies used for the exchange of graphics data between CAD systems. Any possible logic or feature of the object described cannot be transferred to another CAD environment through this method.

Developing a special data-exchange file format, for the building entity, would overcome this restriction. This methodology is already applied in well-known products such as Moldflow, ANSYS etc. The problem of such methods is the flexibility and upgradability in case of incorporating new systems using different data structures.

A second step to integration consists of making the systems talk the same language through a common database, in this case, the "key" to integration is a common database containing virtually all building-object data structures. Every distinct system joining this database will create, store and retrieve only a part of the multi-dimensional features of the Building objects, the information the system is made to deal with. For example, a CADrafting system could create and display only the geometric data of a column or a wall, while a mathematical modeling system could calculate and attach the stress applied on these elements.

At present, the Instituto E. Torroja is carrying out a research program which is aimed at establishing the basic principles of a data model to describe construction, using concepts such as objects, attributes and relations focusing on the design phase and proposing a model for the design process as well. Different existing data base systems such as relational D.B. and hypermedia will be studied, paying special attention to object oriented D.B. which is considered to be the best solution on future.

The third step should be adding a comprehensive user interface. The question to be discussed here is wheter the users, already accustomed to certain systems or environments, would prefer a new special purpose user interface instead of their own good old shell and change their habits in favour of the integration.

The answer is "no, not yet". We have concluded that the users are not ready to accept any changes in their habits, unless someone could prove that substantial benefits could be derived from a possible integration of their systems.

A common user interface will not substantially improve the overall performance of the BD process because the different systems are used by different experts. Unification of user interfaces could benefit only integrated environments (ex. Large companies covering a wide range of BD activities) but again, only in terms of common know-how, training and maintenance. Even in these environments the different disciplines are also carried out by different experts.

7 CONCLUSIONS

AEC CADesign systems, using "real-object" concepts, features and parametrics are not yet found in the market. The Design phase, also called the Concept phase, is where most of the major decisions about style, materials performance and energy consumption takes place. These decisions have the greatest impact on overall Building cost and marketability, and require the most sophisticated design tools. During this phase a variety of alternatives may be explored with the aid of performance simulation and cost-estimating models.

Most CADrafting systems are not well suited to conceptual design tasks. However, given enough support, one may be able to overcome the tools' inherent clumsiness, and turn them into cost-effective aids, by integrating the various

disciplines and technologies able to support the conceptual design phase.

This requires the involvement of databases dealing with objects and cost-estimating databases, Spreadsheet programs to analyze cost data, critical path scheduling programs to develop and redefine schedules and decision milestones, solid-modelling dealing with Building object attributes, surface-modelling tools for the definition of complex plate surface, finite element analysis tools to evaluate stress and heat transfer, personal modelling aids which let Engineers develop "quick and dirty" mathematical models of engineering systems, drafting and animation tools for development of layouts and interior details and many other technologies capable to enhancing the decisions made by the Building designers.

Such systems are being developed separately or are increasingly migrating from mainframes to the Workstation and Personal Computer levels, driven by a major downward trend in the cost of desk-top intelligence and a corresponding upward trend in performance. At the same time as the Workstation CAD revolution is progressing, AI and Expert Systems technologies are changing the expectations of users in AEC area.

If future conceptual CADesign systems can combine the visual glamour which only graphics can achieve, with the intelligence of the decision support systems, then they will ultimately offer the greatest potential in both Building Design productivity and actual construction improvement. Making the right decision in selecting design and construction alternatives, the designer could save considerable design time and results in more durable, marketable and cost-effective construction.

The next maturity step of the Building Design systems' integration will allow a higher degree of integration with the actual Construction systems, that is integration of Application Software, NC machines and Construction Robots.

Now, one of the first objectives should be to create an Integration Environment for Computer Aided Building Design (BD) systems, consisting of:

- a well documented Building Design process Model.
- a reliable conceptual Model of the Building entity, clearly defining the building objects' hierarchy and the corresponding data structures.
- a database management system reflecting the conceptual Model, dealing with objects rather than data.
- common principles for an industrial user interface, meeting the specific needs of the Building designer.

This should improve:

- the quality of the decision-making of the Building designers
- the overall productivity of the BD process
- the management of the BD projects
- the cost/performance ratio and the quality of the actual construction
- the Building maintenance and repair cycles.

8 REFERENCES

1) KORNEL, K., NOWACK, H. (1988) - Exchange of model presentation information between CAD systems - Computer & Graphics, vol 12, pp. 173-180.

2) BOURDEAU, M. (1989) - Structuration des donnes pour una conception technique integree - Proceedings of XIth International Congress CIB 89, París, Theme III, Vol 1, PP 325-341.

3) RECUERO, A. (1989) - Integrated CAD for Buildings - Proceedings of XIth International Congress CIB 89, París, Theme III, Vol 2, pp. 485-491.

4) RECUERO, A. & PROKOPIOUS, A. Integration of Building Design Computer Systems. Management, Quality and Economics in Building. Edited by E&FN Spon. GB. pp. 1549-1557

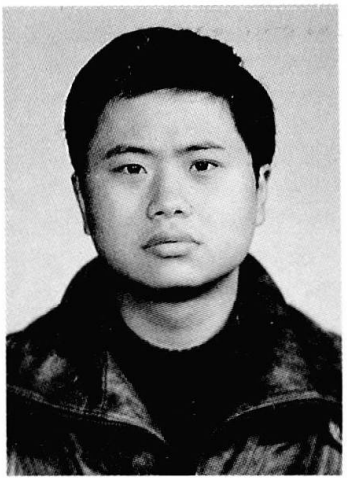
5) TSOU, J; TURNER, J & BORKIN, H. RDBM vs OODBM in Support of Integrated Data Bases for Computer Aided Building Design. CIB W78 workshop on Computer-Integrated Construction, Montreal, Canadá. 1992

6) BJÖRK, B. The Topology of Spaces, Space Boundaries and Enclosing Structures in Building Product Data Models. CIB W78 workshop on Computer-Integrated Construction, Montreal, Canadá. 1992.

Developing Engineering Knowledge Bases
Evolution des bases de données en génie civil
Entwicklung wissensbasierter Datenbanken im Ingenieurwesen

Xiaofeng YANG
Software Eng.
Shanghai Maritime Univ.
Shanghai, China

Kanghen SHEN
President, Prof.
Shanghai Maritime Univ.
Shanghai, China



Yang Xiaofeng, born 1966. got his MS in Computer Science at Shanghai Maritime Univ. His research work is focused on object oriented methodology, knowledge-based systems and Programming Environment

SUMMARY
Engineering standards are widely used in the process of engineering design and manufacture. They are the records of collective and mature knowledge of the profession. The article reports on various software packages developped for application of such knowledge-based systems.

RÉSUMÉ
Les normes du génie civil font l'objet d'un usage très large dans le projet et la construction de structures. Elles reflètent l'ensemble collectif des connaissances de la profession. Le présent article décrit différents logiciels développés pour l'application de tels systèmes à bases de connaissances.

ZUSAMMENFASSUNG
Die Baunormen werden im Bauprozess vom Entwurf bis zur Ausführung vielseitig verwendet. Sie wiederspiegeln das kollektive Wissen des Berufs. Im Artikel werden verschiedene Datenbankprogramme beschrieben, welche entwickelt worden sind, um die Anwendung dieser wissensbasierten Systeme zu erleichtern und damit ihren Wirkungsgrad zu erhöhen.

1. INTRODUCTION

1.1 Engineering codes and KECOMS

Engineering standards or codes are widely used in the process of engineering design and manufacture. They are the records of collective and mature knowledge of the professions. A volume of engineering standards or codes is actually a knowledge base of the engineering profession in a concise literal form.

By now, various engineering codes have been used in CAD/CAM. In (1) (2), engineering codes have been successfully treated as knowledge bases and a knowledge based codes management system, KECOMS was developed.

1.2 The function and composition of KECOMS

KECOMS is a software package which supports engineers to acquire, translate and manage the defined knowledge forms of engineering codes, and therefore to build engineering knowledge bases. With these knowledge bases users can inquire the contents of codes when giving a subject of the request, determine the engineering design constraints according to the design requirements, and verify the conformance of engineering activities according to the related set of codes.

The implemented KECOMS includes following components: (1) a text editor; (2) a graphic editor; (3) a translator of a code knowledge input language - TOL; (4) an information network manager; (5) a classifying tree manager; (6) a data dictionary manager; (7) the inquiry and reasoning system.

1.3 Knowledge representation structures in KECOMS

Various knowledge representation techniques are introduced in KECOMS to represent and prcess the codes knowledge, including decision tables, production rules, classifying trees, frames and procedures. A multi-level codes knowledge structure is adopted. Based on this structure, knowledges in the codes knowledge base are classified into four kinds: the data knowledge, the provision knowledge, the organization knowledge and the application knowledge. Different knowledge takes the different representation structure: the provision knowledge is represented by the decision table structure, the organization knowledge by the semantic network and classifying tree structure and the application knowledge by production rules.

The code processing technique was founded by S.J. Fenves (3)(4). Using decision tables and information networks, he proposed a model as the basis of the standard processing. In KECOMS, the decision tables and information networks are seperated from the code processing. A relevant representation language for decision table, called TOL, was invented. The following is a TOL represented decision table example from a knowledge base of the case "The Tentative Specification of Harbour Work".

Ex.1: a decision table which is represented as a TOL program

```
code jtj22087 5-1-32;
datum L5_1_32 is logical for result;
datum ReInConc(c,1),BiaPull(c,1),Ring(c,1),Tangent(c,1)
      is character*20 for cls;
datum DSCON(c,1) is character*20 for input;
datum  Nz(a,1), Agl(a,1),e0(a,1),rg(a,1) is float for input
```

```
datum K1(a,1) is float for refer ta 3-2-2_t1;
......
datum Betae(a,1) is float for refer dt 5-1-32_1;
ccondition 1: ReinConc=="T";
......
ccondition 5: DSCON=="dscon3";
action 1: (y,y,y,y,y)=>L5_1_32=K1*K2*Nz<=Rgl*Agl*Betae/(1+e0/rg);
end
```

1.4 The coming of OOKSDE

For the complexity of problems in reality, KECOMS takes a mixed knowledge representation structure. It is really a complex task for a system to manage so many kinds of different knowledge structures efficiently. As a result, the knowledge and the processing was not seperated well and some structures took their simple forms in the implemented system. It also results in the limited usage of KECOMS. To solve the problems, OOKSDE (an Object Oriented Knowledge base System Development Environment) is developed.

The initial goal of OOKSDE is to succeed the functions of KECOMS. However, the ultimate goal of OOKSDE is to expand the processing techniques to various engineering professional knowledge processing. The result of our recent researches includes following achievement: an object-oriented knowledge representation method (OOKRM); an object knowledge base design method; an OO knowledge representation language (SMULA) and an object knowledge base system development environment (OOKSDE) (being developed).

2. THE OVERVIEW OF OOKSDE

2.1 The structure and composition of OOKSDE

OOKSDE is a knowledge object base system development environment. The system developed by OOKSDE is a knowledge based system, which contains a collection of various kinds of objects. When combined with the object codes interpreter and the object manager, the system can be on executing. It can perform the operations, such asinquiring code provision content, verifying the engineering design and creating the engineering design constraints etc.. What the developed system may do is determined by the characteristics of the knowledge objects created and developed by the user.

OOKSDE is an integrated environment. It is composed of the following related parts:
(1) a full-screen object editor. It provides the convenient creating, editing and modifying of text contents of knowledge objects.
(2) a knowledge object base manager. It gives the routine processing of objects, such as retrieving, storing, copying and deleting of objects. It also provides mechanism to retrieve and store the data about relations between objects in the object base.
(3) an object inference controller. It performs the operations to control and monitor object evaluating.
(4) a running code interpreter. It performs the real-time interpreting of executable programmed codes in objects. Operating with the object inference controller, it implements the knowledge objects inference. They both make the core of inference in the knowledge object base system.
(5) a user interface generator. It helps to generate the user interface objects for the knowledge base system being developed. The inter-

face objects need to be interpreted by the running code interpreter before going on operation.
(6) a user-friendly interface of environment. It is the interface through which user can do operations on objects and activate the functions provided by the environment.

Figure 1 reveals the components of OOKSDE and their relation in the environment.

2.2 The user interface of OOKSDE

Figure 2 provides the first-stage user interface of OOKSDE.

OOKSDE is designed for developing one and only one knowledge base system during a particular period. More than one knowledge base system can exist at the same time, but only one of them can be active. The process of developing a knowledge base system is a long period task.

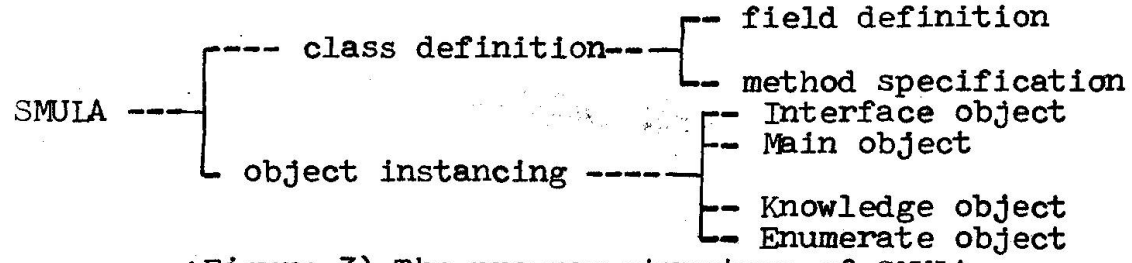
It is supposed that user with OOKSDE can define the class object of knowledge, input the actual knowledge body objects and modify the text content of objects through selecting and entering the (Edit) icon. By selecting the (Compile) icon, objects entered are translated into inner represented forms. Through the (Browser) icon, user can browse the objects stored in system, and can inquire the relation between them. By selecting and entering the (Debug) icon, user can execute and debug the system already built up. The environment also provides the on-line help facilities to give the help information for the system operation, errors and language etc..

3. SMULA, THE KNOWLEDGE REPRESENTATION LANGUAGE IN OOKSDE

3.1 The overview of SMULA

Based on our researching experience in KECOMS, OOKSDE is supposed to be different from other knowledge base system development environments and expert system shells. It provides powerful facilities to handle the engineering knowledge, especially for engineering codes. SMULA, is the knowledge representation and processing language in OOKSDE. In its design and implementation, the following features have considered: (1) the C-PASCAL based language structure is adapted; (2) to represent the special engineering knowledge characteristics, several unique super types are introduced; (3) for processing large scale knowledge bases and introducing reusablity of knowledge , object compiling, template technique and the technique for long period task are introduced in.

A SMULA program can be seperated into two parts, class definition and object instancing. The class definition is for the definition of knowledge object structure. It is composed of two parts. The object instancing is for acquiring actual knowledge body. It is different from each other for the four kinds of objects. See Figure 3.



(Figure 3) The program structure of SMULA

3.2 Some important features of SMULA

Ex.2 : Definition of class object DECISION_TABLE. It is the SMULA represented decision table structure.

```
CLASS  DECISION_TABLE of STRUCT
/*  STRUCT is the super class in OOKSDE  */
{
(1)    def  PNO : string[10];
(2)    def  TEXT_ID : IDENT;
(3)    def  TEXT_V  :  text;
(4)    def  C
(5)      C_begin
(6)        C_expr  :  EXPR;
(7)      C_end   /* condition parts of decision table  */
(8)    def  A
(9)      A_begin
(10)       action_condition  :  array[1..*] of char;
(11)           action_expr  :   EXPR;
(12)     A_end   /*  action parts of decision table */
       /* other method are omitted here  */
       ......
       method  get_dt_result() : value_set;
       /*  evaluate the value of decision table  */
       {
           var_begin
             int   i,j,succ,k;
           var_end:  /* declaration of inner variables */
           for (i=1;i<=STRUCT<-number_of:A;i++) {
              succ = 1;
              for (j=1;j<=STRUCT<-number_of:C;j++) {
                if (action_condition[i][j] != 'I') {
                   k=C_expr j<-Object_value;
                   if (k != 0) {
                      if (action_condition[i][j] != 'Y') {
                           succ = 0;
                           exit;
                      }
                   }
                else {
                  if (action_condition i  j  != 'N') {
                      succ = 0;
                      exit;
                  }
                }
              } /*   for  j  */
              if (succ) return(action_expr  - Object_value);
           }    /*   for i  */
           return(nul);
       }
       method  Object_value() : value_set;
       {
           self  - get_dt_result;
       }
}          /*  DECISION_TABLE   End  */
```

3.2.1 Super types

There are several super types pre-defined in SMULA. They are the expression, the variable and the object statement.

An object of an expression type (EXPR, see Ex.2, (6) (11)) is a string which conform the grammar and semantic structure defined in

OOKSDE. It is needed frequently in processing engineering codes knowledge to represent the mathematical formula. The adoption of expression type makes the acquiring of mathematical formula much conveniently.

An object of a name appears in an expression is treated as an object of a variable type (VARIDENT). The type (VARIDENT) can be re-defined as an object class. This mechanism implements the variable with user defined features.

The object statement type (OBJ_STAT) is also a string. But it must conform the grammar of object activating statement in SMULA. The actual effect is achieved when the name of (OBJ_STAT) type is replaced by its value - the executable object activating statement, and when the statement is interpreted. This operation is called macro substitude in SMULA.

With these super types pre-defined in environment, it becomes a simple task to input the knowledge body objects. Ex.3 presents an actual decision table in SMULA.

Ex.3 : Following is a provision represented as a decision table. The provision c5s1p32 is selected from "The Tentative Specification of Harbour Work". It is instanced according to the object class defined in Ex.2.

```
KB  c5s1p32  of DECISION_TABLE
{
(1)  PNO = "c5s1p32";
(2)  TEXT_ID = "TXT_c5s1p32";
(3)  C  ReinConc == 'T';
(4)  .....
(5)  C  DSCON == "dscon3";
(6)  A (Y,Y,Y,Y,Y),K1*K2*Nz<=Rgl*Agl*Betae/(1+e0/rg);
}
```

3.2.2 Template technique

Template is a special date abstrction mechanism, with three different forms: simple, multiple and nested. Statement (1)-(3) in Ex.2 are simple template. Statement (4)-(7) and (8)-(12) in Ex.2 are multiple template.

The different between simple and multiple template is that the simple one can be instanced just one time but the multiple one can be instanced more than one time. See Ex.3, (1) (2) are the instanced fields of simple template and (3)-(5) are the instanced fields of multiple ones.

With other features, such as repeated definition and default mechanism, template technique greatly simplifies the processing of data abstraction. The language TOL in KECOMS is now in OOKSDE replaced by the class object definitions. Comparing with Ex.1, code provision c5s1p32 in Ex.3 is much more concisely defined.

Reference (6)(7) gives some more examples showing the representation ability of SMULA. Most of commonly used knowledge representation structures can be represented by SMULA.

3.2.3 Object compiling and long period task technique

In OOKSDE, the compilation is based on a single object. The process of building a knowledge base system is treated as a long period task with all temperary statuses are recorded. This makes the developing a knowledge base system can be implemented step by step during a

period of fairly long time.

4. THE PROCESS OF INFERENCE IN KNOWLEDGE OBJECT BASE

The inference of knowledge object base system is the process of evaluating objects. The "sailing" from object to object implements the process of inference. During the process of verifying the comformance of engineering design, the main operation is to evaluate relevant code provisions. In another word, it is to evaluate the objects of relevant decision tables. For short, we consider the evaluating of the object in Ex.3, c5s1p32. To be understood clearly, Ex.4 gives another object represented in SMULA.

Ex.4 Following is an instanced data table object represented by SMULA. The data table is attached to provision c3s2p2.

```
KB  c3s2p2_t1   of  STATIC_TABLE
{
    PNO = "c3s2p2_t1";
    t_data = (1.55,1.45,1.65,1.50,1.65,1.50);
    COL  ((AxPull=='T') (Crook=='T')) & (ReinConc=='T');
    ......
    COL  ((AxPull=='T') (Crook=='T')) & (PressConc=='T');
    LINE  Ladcomb=="design";
    LINE  Ladcomb=="proof";
}
```

When all needed objects are available in the knowledge base, the following relation is automatically built up in the knowledge base system or in OOKSDE. In Figure 4,[<--] means the left one can be evaluated after the right ones have been evaluated.

```
c5s1p32<-- ReinConc (input)
       <-- ..... (input)
       <-- K1 <-- c3s2p2_t1 <-- PresConc (input)
                            <-- ..... (input)
       <-- K2 <-- c3s2p2_t2 <-- Chocond (input)
        ......
       <-- Betae <-- c5s1p32_1 <-- e0 (input)
                               <-- ..... (input)
```

(Figure 4) The relation built up around provision c5s1p32

Suppose the following values are input:
ReinConc = 'T'; DSCON = "dscon3"; Ladcomb = "proof; chocond = "chocond2"; e0 = 5; Agl = 14.5; .....
The evaluation begins with the object c5s1p32. The following is the evaluating chain during inference.

The goal of inference : the value of c5s1p32;
1-1)activate the method Object_value in c5s1p32,
1-2)executing the method in class DECISION_TABLE,
  2-1)evaluate field C, the result is (Y,Y,Y,Y,Y),
  2-2) evaluate field A, that is to evaluate the expression K1*K2*Nz<=Rgl*Agl*Betae/(1+e0/rg),
  2-3)activate the method (expr_value) to get the value of the expression,
    3-1)evaluate the variable K1,
    3-2)activate the method (Object_value) in VARIDENT,
    3-3)activate the method (Object_value) in c3s2p2_t1,
      4-1)evaluate COL field, get the result COL[1] = true,
      4-2)evaluate LINE field, get the result LINE[2]= true,

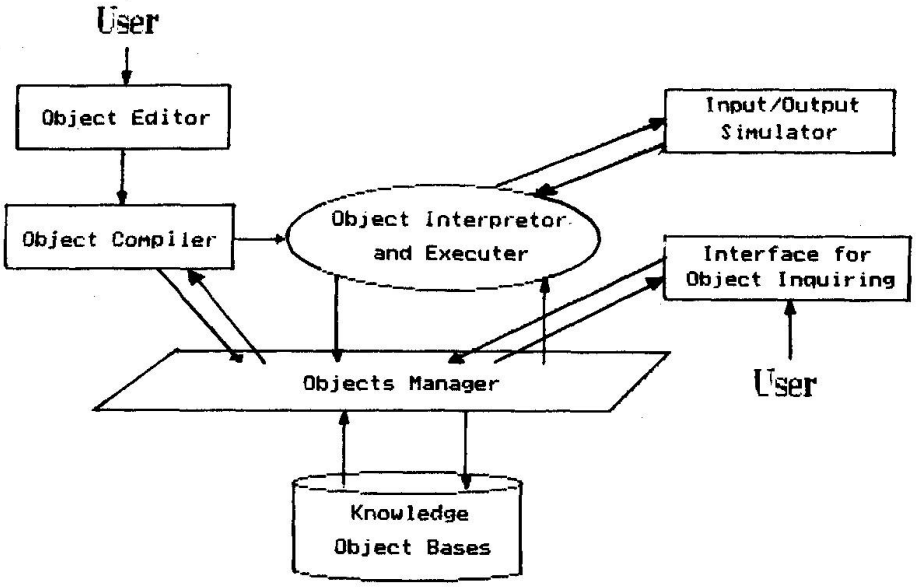
```
      4-3)get the data t_data[1,2] = 1.45,
      4-4)return the value 1.45,
    3-4)get the value, K1=1.45,
    3-5)..... /*  evaluate following variables  */
    3-i)get the value, K2=1.05, Rg1=550, ...
        input value, Nz=4000, e0=5, ...
  2-4)the result is 1, return the value,
1-3)get the value of c5s1p32, the inference process ends.
```

5. CONCLUSION

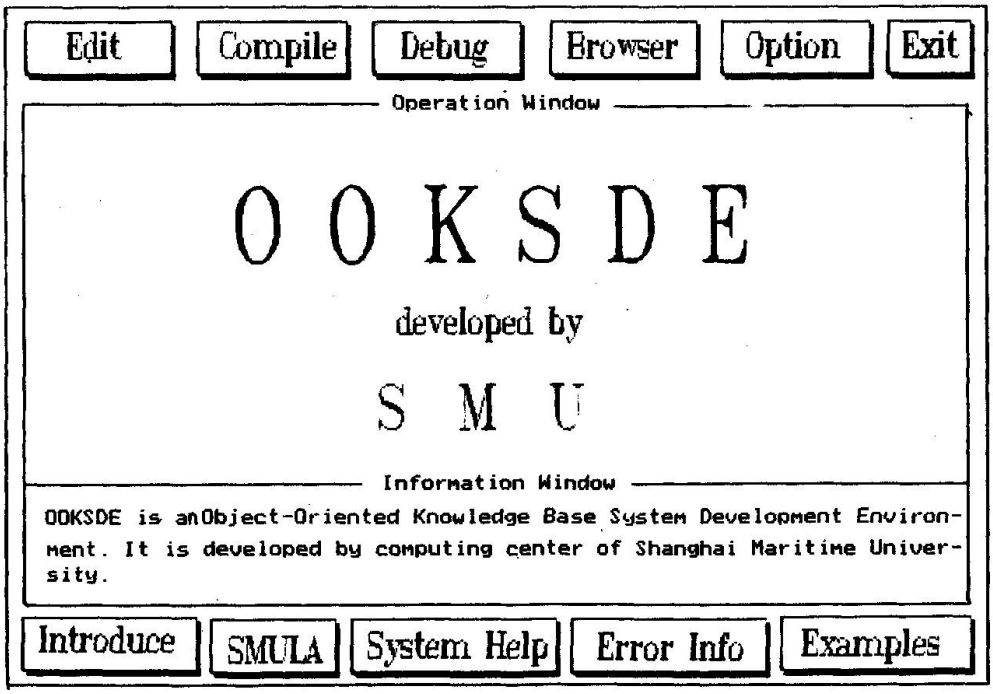
OOKSDE is a development environment for developing common-purposed and large-scaled knowledge base systems. For its unique features of processing code knowledge, it is also an efficient tool to develop engineering knowledge base system. For early development of OOKSDE, it is being implemented using C++.

REFERENCES

1. SHEN KANGCHEN etc, Knowledge Based Engineering Code Management System - KECOMS and Its Interface to Engineering Databases, Proceedings of the International Conference on Expert Systems in Engineering Application, Oct. 1989.
2. SHEN KANGCHEN, etc., ECOMS: A Good Aid for Building Engineering Knowledge Bases, IV-ICCCBE, Tokyo, July 1991.
3. FENVES S.J., Software for Analysis of Standards. Computing in Civil Engineering, New York, 1979.
4. FENVES S.J., Representation of the Computer-Aided Design Process by a Network of Decision Tables. Computer & Structures, Vol.3, p.1099-1107, 1973.
5. YANG XIAOFENG, LI HONG, The Study of OO Knowledge Representation Method and OO Knowledge Bases, Proceedings of CAAI-7, April 1992.
6. YANG XIAOFENG, LI HONG, The Study of Representation of Engineering Code Knowledge, Proceedings of CAAI-7, April 1992.
7. YANG XIAOFENG, SHEN KANGCHEN, SMULA - the Knowledge Representation Language in OOKSDE, To be published on Proceedings of IEEE TENCON'93, Beijing, Oct. 1993.



(Figure 1) The structure of OOKSDE



(Figure 2) The first-stage user interface of OOKSDE

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.

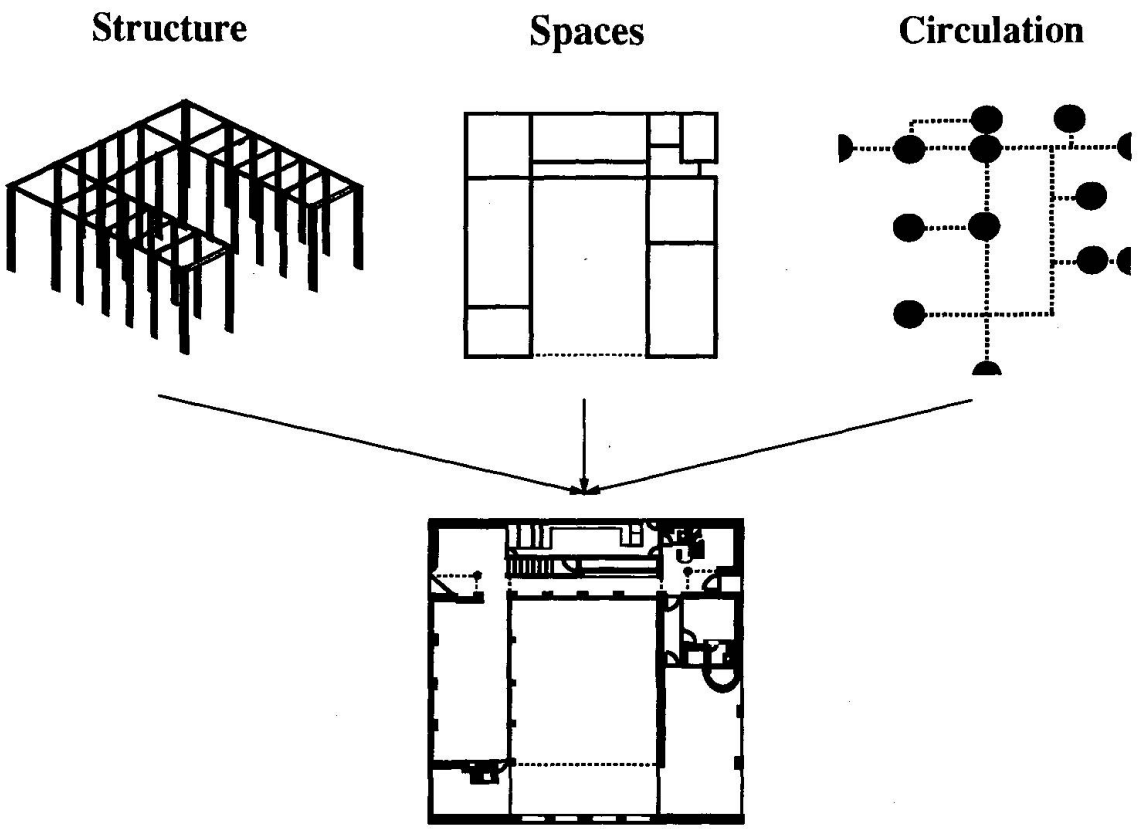


Figure 1: *A building represents an integration of many different abstractions, including structure, spaces and circulation pattern.*

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.

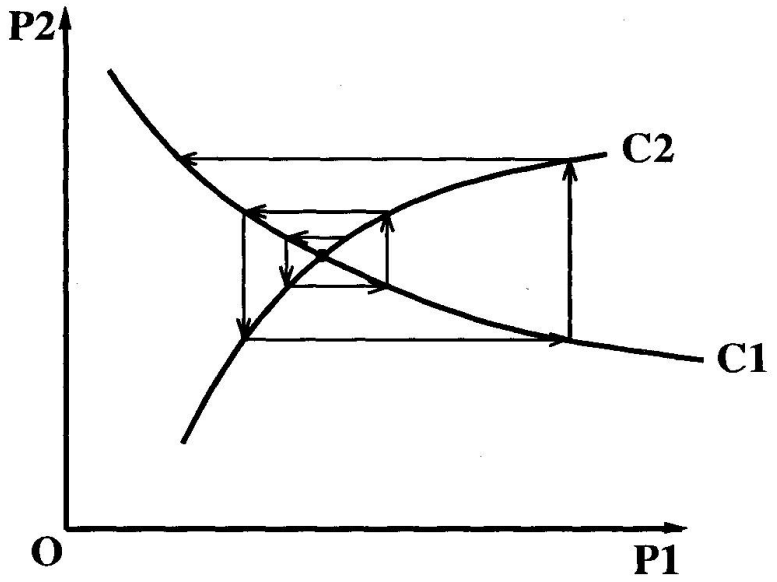


Figure 2: *P1/C1 and P2/C2 represent parameters and constraints in two different but interacting abstractions: the correct choice of the parameter is on the constraint curve. When discrepancies in each abstraction are corrected in isolation, the process may diverge as indicated by the arrows.*

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.

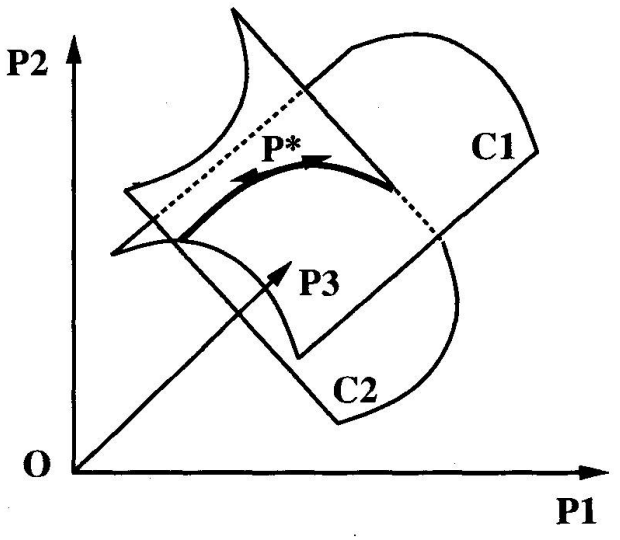
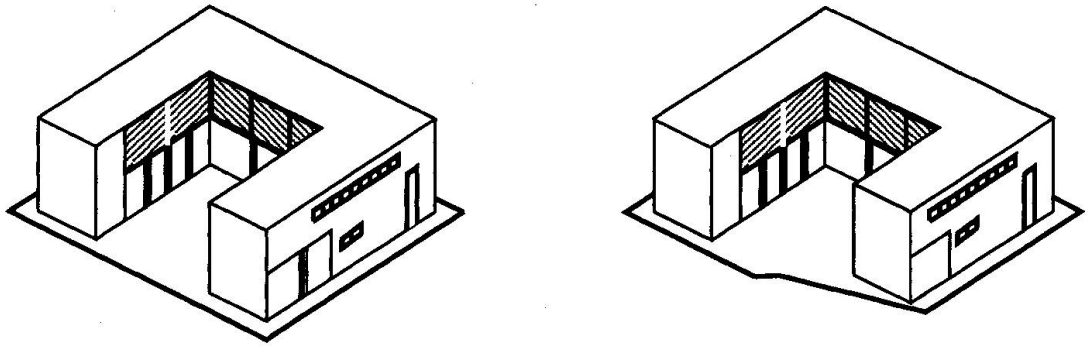


Figure 3: *P1/C1, P2/C2 and P3 represent parameters and constraints of three different abstractions. By defining a new parameter, P*, which maps to positions on the intersection curve of the two constraints C1 and C2, solutions can be found without unstable iterations through selecting a suitable value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

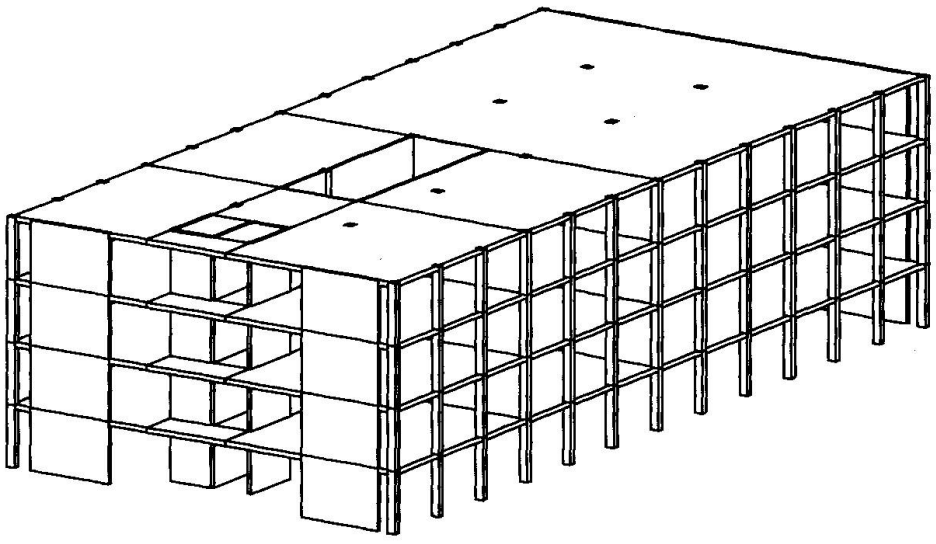


Figure 5: *3D-view of the structure of the INR building.*

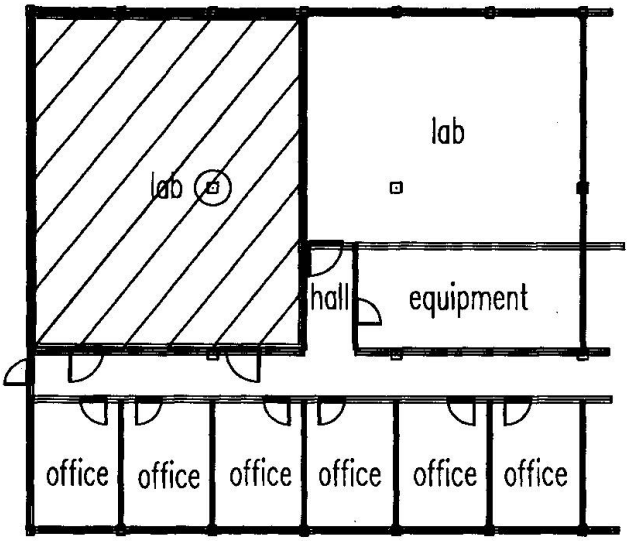


Figure 6: *A partial second floor plan of an earlier building. The hatched area indicates the laboratory space which is used as a classroom in the INR building, recently finished. The blackboard is positioned along the wall on the left hand side of the building.*

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,

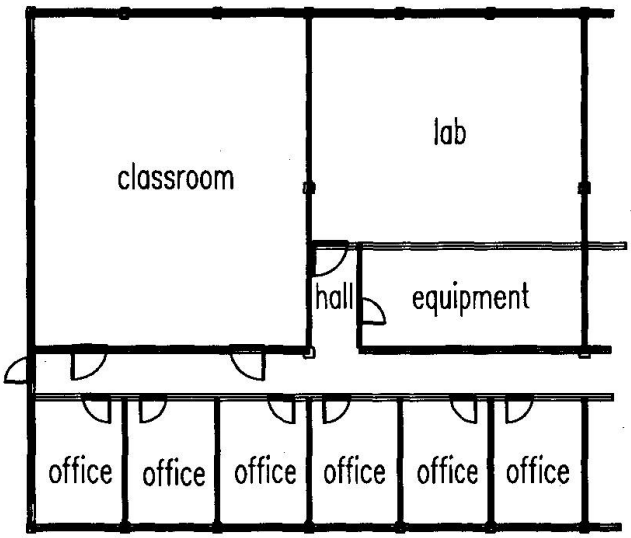


Figure 7: *The new configuration for the second floor of the INR building after structural topological adaptation and after an additional cycle of dimensionality reduction and dimensional adaptation. Constraints related to both architectural and structural abstractions are satisfied.*

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