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

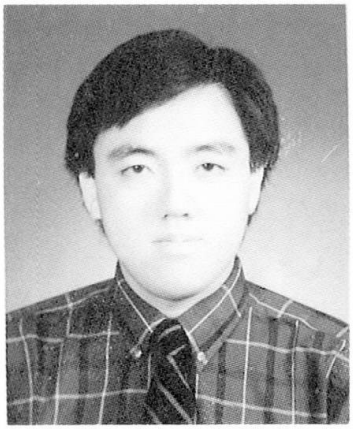
Rubrik: Session 1: Knowledge acquisition and representation

SESSION 1

KNOWLEDGE ACQUISITION AND REPRESENTATION

Issues on Horizontal Integration for Design
Problèmes de l'intégration horizontale dans les projets
Fragen der horizontalen Integration beim Entwerfen

Heah KHENG-LYE
Senior Tutor
Nat. Univ. of Singapore
Singapore



Heah Kheng-Lye, born 1962, received his BSc and MSc in civil engineering from Oklahoma State University, USA. He researches actively in the application of KBES in design. He is currently pursuing his PhD at the Nat. Univ. of Singapore.

SUMMARY
The benefit of having the input of various expert domains at the preliminary conceptual stage cannot be overemphasized. In an integrated design environment the issue is essentially that of integration to allow different expert domains to participate opportunistically in the conceptual design process. A system architecture that permits active communication and effective control of domain activities is proposed. The system architecture is modelled after the blackboard architecture of problem solving and augmented with proper representation models for an overall 'soft control' that allows subdomain design activities and domain-to-domain communication.

RÉSUMÉ
On ne soulignera jamais assez les avantages de pouvoir accéder à divers domaines experts au moment de la phase conceptionnelle des projets. Dans un milieu d'études intégrées, la question est de savoir comment faire appel, en fonction d'un besoin spécifique pour un projet, à ces différents domaines de connaissances. L'auteur propose une architecture de systèmes qui permet d'exercer une communication active et un contrôle efficace dans chacun des domaines d'activité. Cette architecture a été développée d'après la méthode de résolution des problèmes, calquée sur le principe du "tableau noir", et comporte des modèles représentatifs appropriés pour une "commande douce" généralisée; cette dernière permet d'effectuer des travaux d'études dans les sous-domaines et de communiquer entre les domaines.

ZUSAMMENFASSUNG
Die Vorteile der Verfügbarkeit verschiedener Expertengebiete in der konzeptionellen Entwurfsphase können nicht genug betont werden. In einer integralen Entwurfsumgebung stellt sich das Problem, wie solche unterschiedlichen Wissensgebiete nach Bedarf zum Entwurf herangezogen werden können. Zu diesem Zweck wird eine Systemarchitektur vorgeschlagen, die aktive Kommunikation und eine wirksame Kontrolle in den einzelnen Gebieten ermöglicht. Sie wurde gemäss der Problemlösung nach dem Blackboardprinzip entwickelt und enthält geeignete Darstellungsweisen für eine "weiche Steuerung", die Entwurfsaktivitäten in Untergebieten und Kommunikation zwischen Gebieten erlaubt.

1. INTRODUCTION

The existing successful Knowledge-Based Expert Systems (KBES) focused mainly to solve problems of single domains of which they are designed for. Although they are functionally very efficient, they are limited in the perspective of the problems to be solved. This is contradictory to the nature of most practical problems. Many of the problems in reality require more than the input of a single domain. For example, in designing a chair, although the designer may have successfully designed the chair conforming to some design requirements pertaining to the shape, colours, and appearance; the design would further require stress analysis to determine on the suitability of the material chosen for the intended weight of the user and the stability of the chair for normal usage positions. These analyses are apparently beyond the capability of a chair designer who is not expected to possess knowledge of these areas.

Capitalizing on the success of many KBES, researchers worldwide are focusing on integrating several individual KBES for a common problem domain to arrive at probable global satisfying solutions. One of the main area of testing this idea of problem solving engaging integrated KBES is the building design domain. The attractiveness of the problem domain lies in its inherent involvement of different experts.

This paper discusses the building design problem, highlights several integrated design systems, and presents a system architecture addressing a different approach for integrated design environment.

2. BUILDING DESIGN

Building design has been the popular domain of study for integrated design system due to its ill-structured problem nature and the involvement of many professionals on the design of the final artifact. Any building from its inception stage to its completion requires the services of architects, engineers, quantity surveyors, builders, financial advisors, and many others. However, not all professionals are involved at all phases of the building design process. Preliminary investigation requires the services of quantity surveyors and building economists to produce feasibility study for the building project. At the conceptual design stage, basically the architects and the engineers will work together to conjecture probable physical descriptions of the building. Their duties will carry over to the preliminary and detail design phases where the detail analysis of the design will be carried out to eliminate any problems and for the production of final design description. During the construction phase, builders as well as project planners will be employed.

Effectively the entire design process can be broken down into the following three stages:

1. **Conceptual Design.** The building as a whole is synthesized with the available information and design brief. Major components and concepts are developed at this stage. Work at this phase is subject to revision on the detail implementation, but majority of the decisions on the final product would have been made.

2. **Preliminary Design.** Professionals will attempt to study the implications of various

plans for the building. This could involve studies of different structural schemes, mechanical systems, electrical systems, overall building configurations, and others.

3. **Detail Design.** The major concern at this phase would be the production of details for the actual construction of the artifact. Many aspects of the design would have been finalized and any modifications to the design would be costly to rectify.

From the above descriptions of the building design phases, it can be seen that at the conceptual design phase the active involvement of various professionals is crucial to a successful design. The participation of various professionals can be shown schematically in Figure 1.

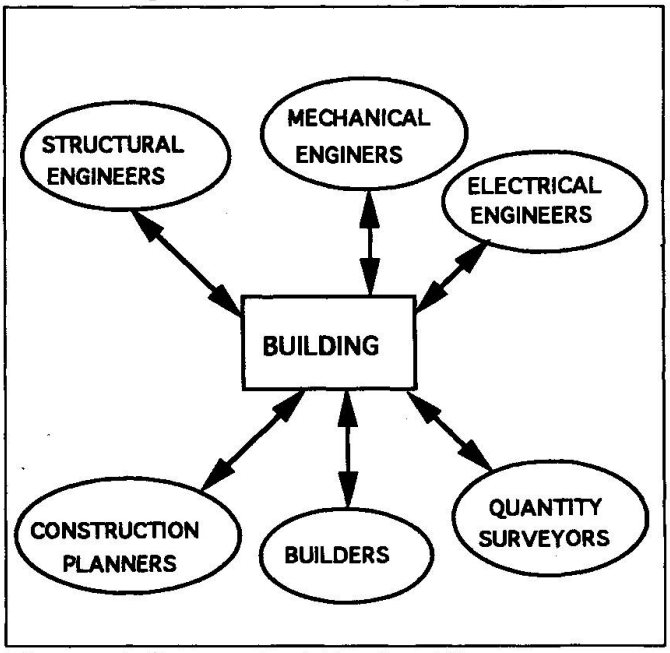


Figure 1 Conceptual Design Phase

With the success stories of KBES in solving problems of specific domains, researchers are aiming to integrate different expert systems to arrive at the globally acceptable design solution for a similar artifact. Each of the expert systems is seen to be effective in offering partial solutions to the whole design problem within the areas of their expertise. This approach requires a process of breaking down the design task into subtasks and allocating the subtasks to the appropriate KBES. Next section highlights some of the integrated design systems and the different approaches adopted by these systems to integrate various KBESs to design.

3.0 INTEGRATED DESIGN SYSTEMS

3.1 Different Approaches to Integrated Design

Jens-Pohl et al.[1] proposed an ambitious integrated design environment which attempts to integrate project program and databases of design information within an intelligent computer-based design environment. The environment consists of basically three components: an intelligent CAD DBMS; an Expert Design Generator; and, a Multi-Media Presentation Facility. The expert design generator essentially is made up of modules of Intelligent Design Tools that are coordinated by a Blackboard control system. The intelligent design tools perform the tasks of mechanical system design, structural system design, and other essential design tasks. While the multi-media presentation facility assists the user in visualising and understanding the design, the DBMS is designed to house the information on prototype buildings and site information. The entire project is aimed at a total computer assisted design environment, where the computer plays the central role in the development of the design solution.

Another integrated design environment that utilizes the blackboard model of problem solving in coordinating design solution process is the Integrated Building Design Environment (IBDE) at Carnegie Mellon University[2]. The environment consists of seven distinct components: ARCHPLAN, CORE, STRYPES, STANLAY, SPEX, FOOTER, and CONSTRUCTION PLANEX. Each of the KBES performs specific functions where ARCHPLAN tackles the development of preliminary design brief; CORE aims in the planning of the service core; STRYPES performs preliminary structural system configuration; STANLAY lays out the structural systems and performs preliminary structural analysis; SPEX continues with the preliminary design of the structural elements; while FOOTER designs the foundation system; lastly CONSTRUCTION PLANEX will assist in the construction planning.

The seven processes (as named) are actually standalone KBES that can perform their tasks efficiently without acknowledging other KBES provided appropriate design information is available. Some of the processes or KBESs are actually developed before the IBDE while others are specifically developed for the environment. Their participation in IBDE is coordinated by a status blackboard and controlled by a controller. Though a comprehensive environment which addresses the unique characteristics of building design, IBDE essentially, with its processes are envisaged to be located in different workstations, tackles the building design problem in a linear, sequential manner.

The two examples have illustrated the popular approaches normally used for the development of integrated design systems. In one case a central database is used to facilitate the exchange and the communication of design information from one design system to another and with one another within an environment. The design systems are independent of each other in terms of addressing the design problem and the participation in the design process. This is the approach adopted by Jens-Pohl et al. Often, a more efficient integrated system maybe derived for a specific problem domain through incorporation of design processes and directly addressing the design system. IBDE was planned to be such a system. Nonetheless, the integration of IBDE is essentially linear and sequential with the domains integrated discretely. This method is useful at the detail design phase but has not addressed the actual design scenario at conceptual design phase.

3.2 Typical Integrated Design Architecture

Previous section has illustrated two prototypes of integrated design systems for building design. Generally, an integrated design system consists of the following four major components:

1. *A Database*. This database serve as the storage for the global descriptions of the design artifact represented hierarchically. There may be more than one database if the system calls for it.

2. *A knowledge base*. The knowledge base consists of the knowledge of a particular domain. There should be more than one domain in an integrated design system. Each of the knowledge base may adopt different knowledge representation model. Primarily they provide design heuristics for the domains they are responsible for.

3. *An inference engine.* Inference engines are designed to work with the representation model of each knowledge base. The inference engines are to reason on the problem and to arrive at solutions using the knowledge base.

4. *A blackboard.* There may be more than one blackboard in an integrated design system with each blackboard performs specific functions. Normally a control blackboard is designed to oversee the activation of different knowledge bases. It will identify which knowledge base to activate based on the conditions it has and the status of the problem appropriate for that knowledge base to contribute to the solution process. In some cases a status blackboard is employed which is responsible for the status of processes.[3]

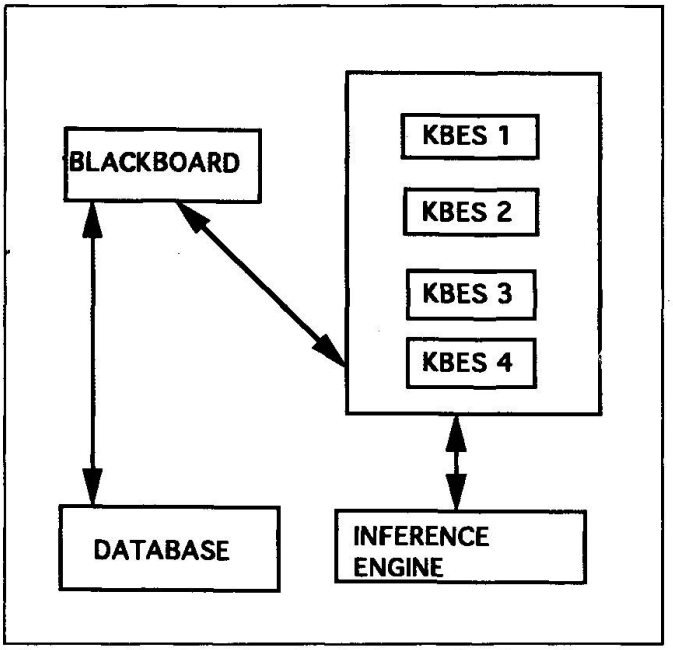


Figure 2 Typical Integrated Design Architecture

Figure 2 shows the general architecture of an integrated design system.

4.0 EFFECTIVE DESIGN INTEGRATION

4.1 Issues for Effective Design Integration

Instead of employing integration at the detail design stage, it is arguable that the most suitable phase for the involvement of various design experts in the domain of building design is the conceptual design phase. Decisions that affect approximately 80% of the total cost of a building are often made at this conceptual design phase.[4] It is conceivable that the large proportion of crucial design decisions at such an early stage in the design of a building would results in costly remedy later on should errors occur. An integrated design system should emulate the conceptual design phase during which professionals from different disciplines make their expertise available to assist in the formulation of the design artifact instinctively and opportunistically.

The integration of various expert domains at the conceptual design phase will be different from the integrated design systems mentioned in the previous section. Notably are the following differences:

1. Instead of a linear, sequential integration, a simultaneous and horizontal integration should be implemented. This would allow different domains to contribute to the overall design as the design information is becoming sufficient in their domains. Each of the domain will be aware of the status of the intermediate

partial design solutions and will act accordingly.

2. A lower level of control can be realized through the delegation of 'soft control' to the individual expert domains. The advantage of such setup would be to allow the individual domain to trigger the execution of other appropriate domains to contribute to the design solution process; then the first domain will act on the generated design information.

3. Communication will be divided into two levels. The first level will be the global design status and the second level is at the domain level. Global level communication is defined as that between the global objects base and the blackboard. Domain level communication is activated through the proper structuring of design heuristics and the provision of rich knowledge representation model.

The emphasis of the horizontal integration approach is the active, spontaneous, and opportunistic response of the expert domains. A system with the capability to allow such integration will serve well the problem of the conceptual design phase; that of lack of initial design information, and spontaneous contribution to the problem solving process. Instead of waiting for design information to be made available, a domain will seek out next proper domain to contribute to the design information thus enhancing the original design information to enable it to react to. This is representative of the actual design scenario as described in previous section.

Clearly, the functionality of such a system is beyond the problem solving paradigm advocated by blackboard models of problem solving. Typically blackboard systems are sequential though the knowledge will be rich enough to reason on the status of problem and to determine the next course of action, it remains that the system will access a 'central command post' which is not versatile enough for the problem solving approach desired in the previous paragraphs. Nonetheless it offers a good starting ground for the system architecture described in the next section that attempts to address the condition of spontaneous and opportunistic response of expert domains.

4.2 Integration with Active Participation

This section is structured to present the proposed system architecture. A few issues must be addressed with respect to the discussion of an integrated system. Figure 3 shows the overall system architecture with the necessary components. Essentially the system consists of a control blackboard, a global objects base, and various knowledge bases.

1. ***Global Objects Base.*** The major deviation from the conventional blackboard model of problem solving is the representation of the problem domain using objects and the development of hierarchical structure of these objects to represent different levels of partial solutions. The object representation allows the mapping of the global objects with that of the individual knowledge bases easily. This facilitates communication among knowledge bases as well as with the global objects. This represents a major difference between hierarchical representation in a status blackboard and the global objects base. Where a status blackboard

records the status of global solutions and make it available for the control blackboard to decide on next course of action; a global object will determine independently of the next domains which will contribute to the current design information. Status of the global objects can be determined readily by a control blackboard. Hierarchical structure of the global objects also allows status report of the partial global solutions to the control blackboard.

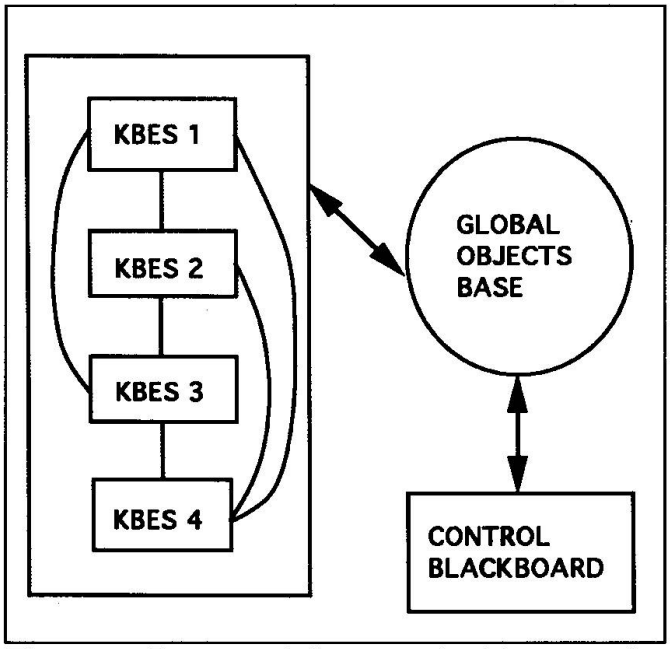


Figure 3 Proposed System Architecture for Integrated Design

2. ***Control Blackboard.*** At the core of the system architecture is this control blackboard. Rather than controlling the actions and the recording of status of solution process, the blackboard determines domain objects status, maps the domain objects to the global objects, checks on the global objects base to determine on their status and translates that into universal partial solutions to compare with the global solutions. It has no active function of activating appropriate knowledge bases for further actions. But it has the important task of determining the degree of completion of the global solutions. As communication and activation of knowledge bases are delegated to each domains, the control blackboard reasons at the global level.

3. ***Knowledge Bases.*** Each knowledge bases is composed of its own inference engine and expert knowledge. Object representation is used to ensure uniform representation and to enable mapping to the global object base. Objects at the domain level of a knowledge base, which is augmented with rules to guide on the solution process, decide on whether the design information at any time of the solution process is sufficient for it to contribute to the global solution. If not, the domain objects will determine on what are the design information it needs for it to contribute to the current solution status. It communicates directly with other domains that can contribute to its collection of design information for it to take action.

4.3 Controlling Solution Process

Unlike most integrated design systems which have in the heart of the systems blackboards that watch over the solution process, keep records of activation of knowledge sources (a term used widely in blackboards which essentially means knowledge bases), and determine the next course of action and the knowledge source to activate, the proposed system, described in previous section, delegates the essential control of the solution process to the individual domain objects.

This translate into the requirement for an effective implementation plan. The knowledge bases will have to be designed conforming to the global objects in the global objects base. This is seen as necessary in order to have effective communication between the global objects and the domain objects. A derived benefit of the implementation is the uniform representation of the objects in various domains; hence, enhancing the domain level communication.

4.4 Object Representation

Object representation is employed in both the global objects base and the knowledge bases because of the flexibility of the model. It allows hierarchical structuring of the solutions and permits the eventual mapping from domain to global objects. Coupling with rules, the objects can effectively determines the next course of action even at the domain level. Ease of expanding the object descriptions also was taken into consideration for choosing this knowledge representation model.

The global objects base can be expanded incrementally as more knowledge bases are added to the system. These knowledge bases will inevitably introduce new objects to describe their domain objects which would form additional descriptions for the global objects base.

5. DISCUSSIONS AND CONCLUSION

An attempt is described in the paper on how to effectively introduce opportunistic and spontaneous style of participation of several expert domains in an environment. Experience from the development of such a system has called for careful design and implementation of the global objects base and the blackboard.

Knowledge representation model has been given very special attention in the development of the system, it is seen as a crucial and very complex task. The proposed system has not taken advantage of the clear and explicit representation of the solutions as offered by the conventional blackboard systems. The detail implementation of blackboard is outside the scope of current discussion.

However, active participation and the activation of appropriate domains at the domain level is achieved through the sacrifice of this obvious advantage of blackboard models of problem solving. In addition the proposed system has advocated a representation using objects which can be very effective in hierarchical structuring of the partial solutions. In spite of, it has incorporated a blackboard to check on the status of the global solution objects.

There remain several issues need to be taken care of for effective implementation of the system architecture. Hardware constraint is a major factor considering the speed of execution since the knowledge bases are expected to be resided within a single computer system. System memory requires for such an environment will also be unusually demanding. Complexity in structuring the knowledge and objects cannot be overemphasized too.

REFERENCES

1. POHL J., CHAPMAN A., CHIRICA L., HOWELL R., and MYERS L., Implementation Strategies for a Prototype ICADS Working Model. Design Institute Report, California Polytechnic State University, San Luis Obispo, December 1988.

2. FENVES S.J., FLEMMING U., HENDRICKSON C., MAHER M.L., and SCHMITT G., Integrated Software Environment for Building Design and Construction. Computer-aided Design, Vol. 22, No. 1, Butterworth & Co., Jan/Feb 1990.

3. ENGELMORE, R. and MORGAN T., Blackboard Systems. Addison-Wesley, 1989.

4. FERRY D. J., BRANDON P.S., Cost Planning of Buildings. 5th ed., Granada, 1984.

Automatic Knowledge Acquisition and Economic Evaluation
Acquisition automatisée des connaissances et valeur économique
Automatisierter Wissenserwerb in Wirtschaftlichkeitsstudien

Lianzi WANG
Associate Professor
N. Jiaotong Univ.
Beijing, China

Yang XU
Lecturer
N. Jiaotong Univ.
Beijing, China

Lianzi Wang, born 1933, graduated at Tangsban Jiaotong Univ. 1955. At Northern Jiaotong Univ. she is responsible for automatic application in design of railway by computer.

Yang Xu, born 1961, got his BS degree in computer science at the Beijing Computer Inst. in 1985, and his MS degree in engineering economics at the Northern Jiaotong Univ. in 1991.

SUMMARY
The paper is intended to describe the method for automatic acquisition of knowledge by way of a neural network to be applied in the economic evaluation of railway construction or upgrading projects. Based on the features of knowledge structure and knowledge system in the field of engineering economics, the necessary analysis and modification of a typical neural network model, with regard to the structure and algorithm are carried out, leading to a package of networks and algorithm suitable for engineering economic analysis. This paper also describes in depth the new type expert system which combines the neural networks with the expert system in this regard.

RÉSUMÉ
L'article propose une méthode d'auto-apprentissage des connaissances par le biais d'un réseau neuronal, développé pour les calculs de rentabilité dans les projets de construction ou de modernisation de chemins de fer. A partir des caractéristiques de structures et de systèmes de connaissances dans le domaine de la rentabilité, un réseau neuronal typique est analysé et modifié tant en structure qu'en algorithme, jusqu'à faire apparaître un progiciel de réseaux et d'algorithmes pouvant être appliqué dans les calculs de rentabilité. Les auteurs décrivent en outre un nouveau système expert dans lequel les réseaux neuronaux ci-dessus ont été mis en application.

ZUSAMMENFASSUNG
Es wird eine Methode des selbsttätigen Lernens in einem neuronalen Netzwerk vorgestellt, die für Wirtschaftlichkeitsberechnungen von Eisenbahnprojekten entwickelt wurde. Aufgrund von Struktur- und Systemmerkmalen des ingenieur-ökonomischen Wissens wird ein typisches neuronales Netzwerk hinsichtlich Struktur und Algorithmus analysiert und modifiziert, bis ein Paket zum Einsatz bei Wirtschaftlichkeitsberechnungen entsteht. Der Beitrag beschreibt ferner ein Expertensystem, in das solche neuronalen Netzwerke implementiert wurden.

1. INTRODUCTION

Economic evaluation is the essential factor in the overall assessment of a railway construction project. It involves not only the quantitative computation but also qualitative experience. For the latter case, it is rather difficult to describe by direct use of a mathematic model. But anexpert system can fulfil the task of both, i. e. to carry out the qualitative analysis based on the experience of the expert and the quantitative computation through the procedural knowledge, which can be best suited to the economic evaluation.

There are many evaluation methods available for the economic evalualtion of the railway construction projects. The diversification of the methods often leads to a lot of troubles to the summing—up of the experience of the experts as well as the review of the laws [1].

Neural network is a functional structure of wide applications. It can simulate systems with very complicated properties. By using computer as a means to substitute manual labor and based on standard algorithms, it can summarize and acquire the necessary knowledge and laws through the immense existing samples [2]. The combination of neural network with expert system for the economic evalution of railway construction projects can play a great role in the simplification of the system reasoning mechanism and the constant updatiang of the knowledge base.

2. ANALYSIS OF DOMAIN KNOWLEDGE STRUCTURE [2]

Presently there are available many types of neural network. Only careful analysis is made to the various neural networks, can a suitable neural network model be selected.

2. 1 Classification of domain knowledge

The domain knowledge can be classified into the following according to the system processing methods:

2. 1. 1 procedural knowledge

This kind of knowledge generally involves certain defination formulars, theorectical equations or emperical formulars. It is represented by procedure or functions in the programs.

2. 1. 2 Reasoning expert experience knowledge

It is the typical type of expert experience knowledge. It is represented generally by the way of " IF. . . THEN. . . "

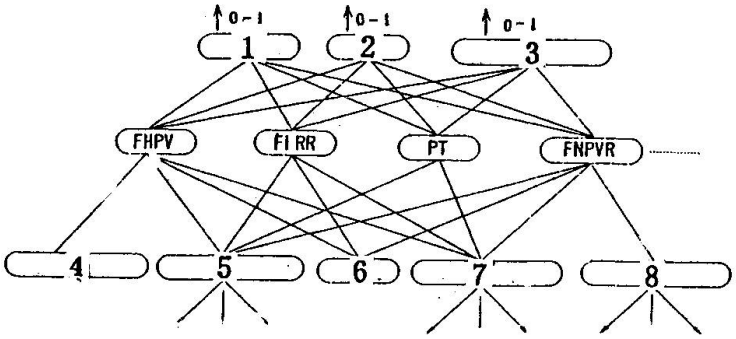
2. 1. 3 Evaluation type expert experience knowledge

This type of knowledge is generally based on a set of evaluation criteria with the support of the expert experience for the selection of the most suitable conclusion. Different from the previous one, it is rather by refering to the existing evaluation samples or the results of analysis through certain mathematic methods than the direct acquisition from the experts for the summing—up, analysis and conclusion.

2. 2 System structure of domain knowledge

And the economic evaluation of a railway construction project must be based on the objective facts as well as the objective environment to determine the input and output of the project. Then a series of evaluation criteria is obtained accordingly, by which the final evaluation conclusion is drawn.

It is seen that we can draw such a conclusion; The domain knowledge structure for the economic evaluation of the railway construction projects is that of a hieracy structure, which can be shown in Fig. 1.



1. Evalnation conclnsion
2. Lmmediate construction
3. Postponement not necessary
4. Rate of discount in dynamic recovery
5. Cash in flow
6. Calculation period
7. Cash out flow
8. Present value of investment

Fig. 1

3. IMPROVED B—P MODEL FOR ECONOMIC EVALUATION

In view of the above mentioned domain knowledge structure, we selected a similarly—structured neural network, i. e. the B—P (Back—Propagation) Model and made certain necessary modifications accordingly so as to best suit the actual conditions.

3. 1 The B—P Model is a multi—hieracy network structure [3][4][5][6].

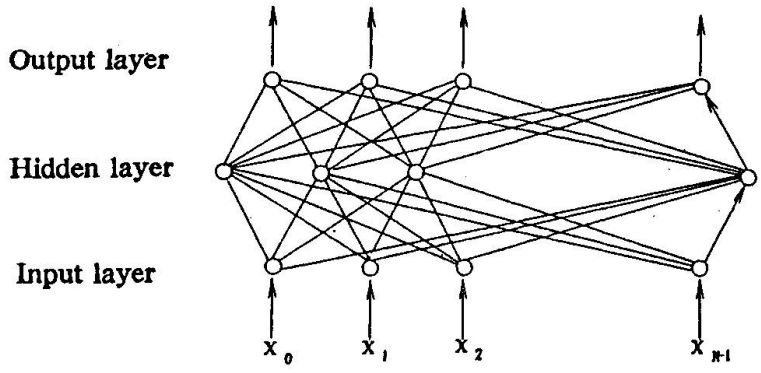


Fig. 2

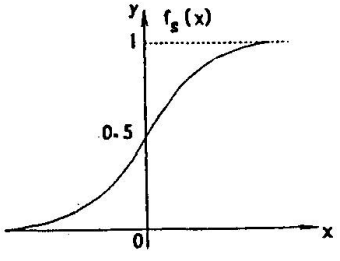


Fig. 3

Perfect unidirectional connection is done between adjacent layers (Fig. 2). It has input and output nodes as well as hidden nodes. The input value has to be transmitted to the hidden nodes first for processing through the activation function and then the output information is further transmiteted to the next hidden nodes layer by layer until to the output nodes. (The hidden nodes probably consist of more than one layer.) The output value is formed at the output nodes. The activation function se-

lected for the various nodes are shown by the S—functions in Fig. 3.

$$f_s(x)=\frac{1}{1+\exp(-x)}$$

The learning process of a network is that in which the error is reduced while back—propagating. The error function for the Pth sample is shown as follows:

$$E_p=1/2\sum(t_{pj}-O_{pj})^2$$

where tpi, Opj are respectively the expected output and the actual output for the Pth sample. The purpose of the weight correction is to reduce the error function. Therefore we selected the gradient descent method of the optimization theory so as to reduce the weight value of the network in the direction of Ep gradient.

$$\Delta PW_{ji}\infty\frac{\partial E_p}{\partial W_{ji}}$$

The computation for the weight correction is shown as follows:

$$\Delta PW_{ji}=\eta\delta_{pj}O_{pj}\quad \text{When Opj is a hidden node, } \delta_{pj}=f_s'(net_{pj})\sum\delta_{pR}W_{Rj}$$

And when Opj is a output node, $\delta_{pj}=f_s'(net_{pj})(t_{pj}-O_{pj})$

For the above equations, Wji is the weigh value for connection between the ith node and the jth node of the upper layer. Opj is the output of the jth node, in which $net_{pj}=\sum W_{ji}O_{pi}$ is the information sum received at the jth node, δ_{pj} is the output error at the jth node, and $\eta>0$ is the gain.

For this type of models, due to the existence of the hidden nodes which are equilavent to the introduction of immense adjustable parameters in the model, the capability and flexibility of the model is greatly enhanced. The following theorem is developed:

If the hidden nodes could be set up arbitrarily, then the network with the three—layer S—function may approximate to continuous arbitrary functions for any accuracy[5].

From the above mentioned, it can be seen that the B—P Model can satisfy the knowledge structure shown in Fig. 1 in general. However necessary modifications are required.

3. 2 Modification of B—P model[7][8][9]

3. 2. 1 Modification of the network structure

a. In the B—P Model as shown in Fig. 2, the unidirectional perfect connection is done between adjacent layers, but some are not neccessary at all and can be deleted completely according to the actural conditions in its application in the economic evaluation.

b. There are many types of input values for the initial samples. If all the values are included into one neural network it would increase the complexity and also reduce the learning speed of the neural network and it would lead to difficult conclusion of the evalutions. Therefore, the neural network should be structured properly, the training, conducted separately, and the outcome information,

plications.

c. The number of hidden nodes can be adjusted for the above mentioned various neural network in case of need and it is not necessary to set up hidden node layers for linear problems[8].

As mentioned above, the theorem can ensure that the three layer B—P Model can simulate the arbitrary continuous functions but as a matter of fact, it is rather difficult to accurately conduct the computation simulation due to the built—in shortcomings. Hence, in this system, the B—P Model is not intended to represent the various procedural knowledge but instead, it is used to represent the reasoning expert experience knowledge and in particular the evaluation type expert experience knowledge. This is more practical and realistic for the increasing of the operational efficiency of the system.

3. 2. 2 Improvements of the algorithm of the B—P Model

a. In the knowledge network for economic evaluation, there are not only data input nodes but also status input nodes. For the latter case, the status has to be quantified first, i. e. to represent different status by corresponding quantities.

b. It can be seen that the sensitivity area of changes for activation function and its derivative is rather small and is only in $[-4,4]$ as shown in Fig. 4. Therefore, the sample value should be located in this area as far as possible so as to better adjust the weight valaue (for sample of faster changes, this area can be reduced).

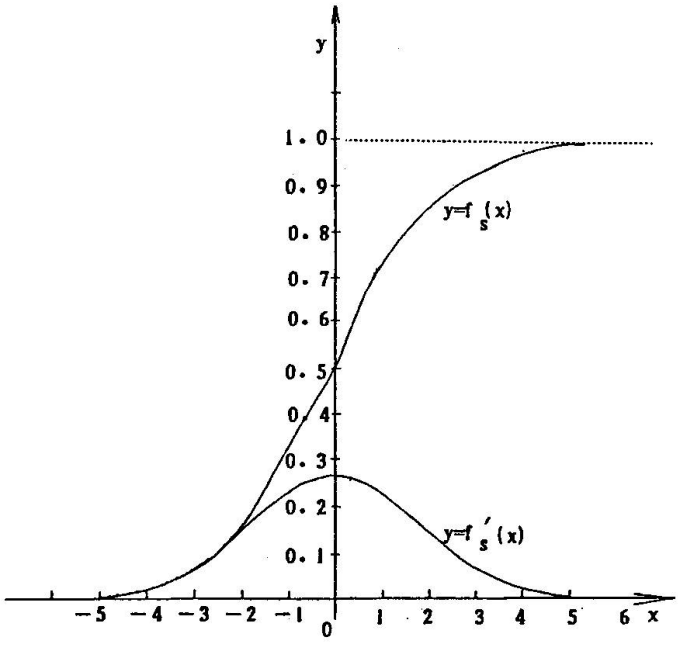


Fig. 4

Assuming that there are L given samples, X_i^R, $i=1,2\ldots,L$, $R=1,2\ldots,$ the following conversion for X should be conducted:

$Y_i^R = \frac{8(X_i^R - b)}{a-b} - 4$ Where $a = \max\{x_i^R\}$, $b = \min\{x_i^R\}$

By this way the input value of the sample is compressed to the area of $[-4, 4]$.

For the output value of the network, a threshold of $0 < \theta < 1$ should be selected so as to keep the output value being 1 when $f_s(X) \geqslant \theta$ or zero.

In view of the above mentioned statement, the algorithnm of the improved B—P Model is shown in Fig. 5.

4. COMBINATION OF NEURAL NETWORK WITH EXPERT SYSTEM[11], [12], [13]

Since the neural network possesses strong knowledge acquisition capabilities and system simulation functions, there have developed quite several expert systems structured by neural network. However, it is the author's view that these systems are mere traditional expert system represented in the form of neural network hence bringing forth some certain problems in actual applications.

a. The interpretation function of the system is rather weak and it is suitable only for certain training samples and lacks representativity in many cases.

b. Due to the inadequecy of the training samples the system sometimes can not solve the desired problems or it will generate more than one conclusions.

c. It might cause the complete abortion of the existing expert system and, in particlular, the valuable knowledge base.

The above mentioned three problems, especially the first two, have been solved satisfactorily by using the traditional expert system. But on the other hand, it is difficult for the traditional expert system to realize the automatic acquisition of knowledge and parallel processing. However, these shortcomings can be well coped with by the neural network. A proper artificial intellegent system should be a traditional expert system to be combined with the neural network. This newly—formed artificial intellegent system should be composed of and structured as shown in Fig. 6.

The traditional expert system is explained in [12].

In the neural network system, the Neural Network Knowledge Base (NNKB) is actually a set of improved B—P neural network Models, which is constantly updating with the running of the system. The neural network is trained in the training module by using the samples and the improved B—P algorithm. The solver is used to get the conclusions through the use of knowledge base and the inputs as well as the working method of the B—P Model. The verification module is used to verify the reasonability of the conclusions.

This new type of intellegent system is design to operat in such a way that each issue must go first to the neural network for solutions. The outputs might be in one of the following three patterns:

1. Only one output from the output node is greater than the limit value. In this case, the system will interprete and output the conclusion.

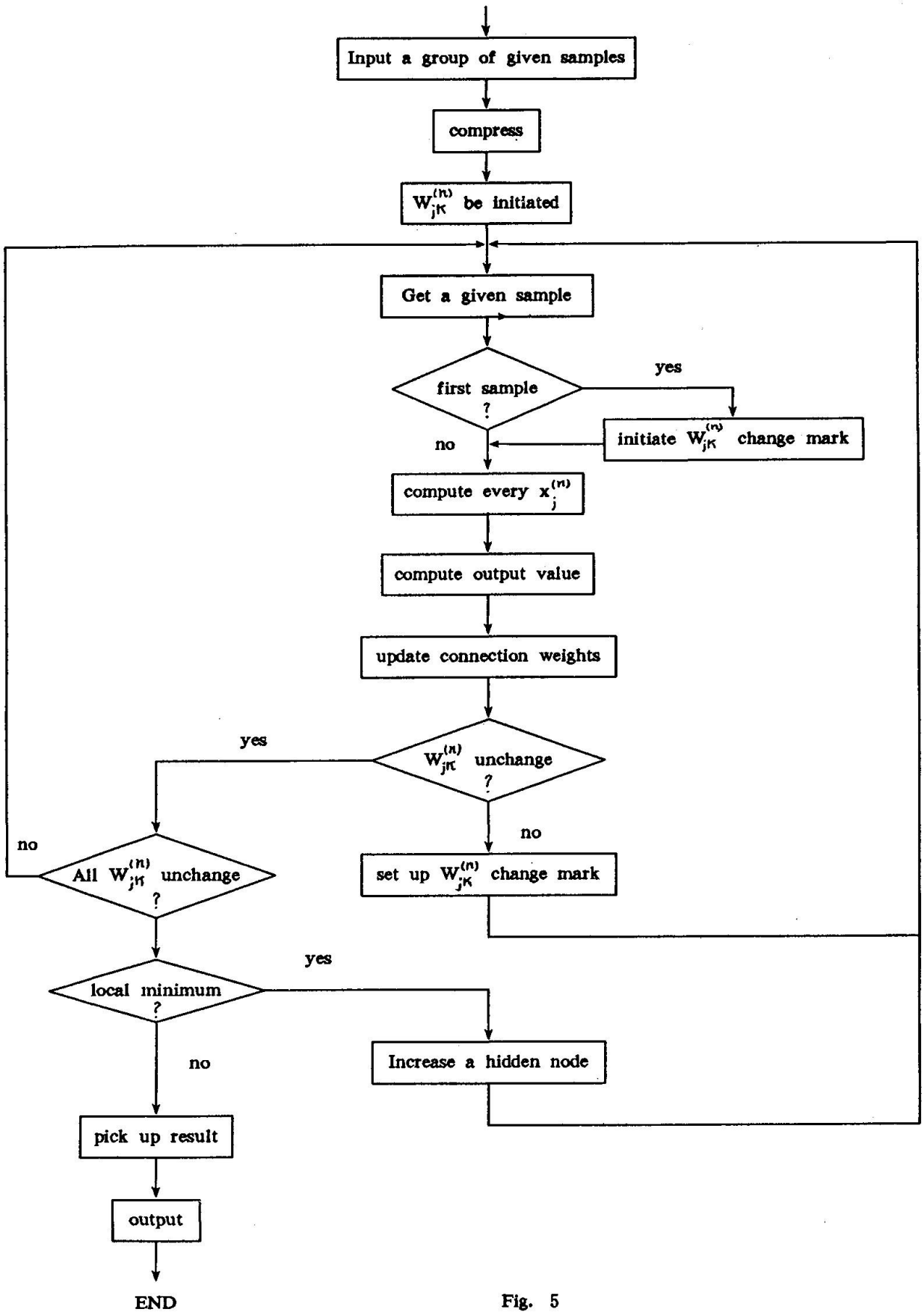

Input a group of given samples
compress
$W_{jK}^{(n)}$ be initiated
Get a given sample
first sample ?
yes
initiate $W_{jK}^{(n)}$ change mark
no
compute every $x_j^{(n)}$
compute output value
update connection weights
$W_{jK}^{(n)}$ unchange ?
yes
no
All $W_{jK}^{(n)}$ unchange ?
no
set up $W_{jK}^{(n)}$ change mark
local minimum ?
yes
no
Increase a hidden node
pick up result
output
END

Fig. 5

2. Several outputs from the output node are greater than the limit value. In this case, all the outputs can be treated as candidates for conclusions and then the inference method is used for deriving the final conclusion.

3. No output from the output mode is greater than the limit value, In this case, only the traditional expert system is to be used.

For the second and the third cases, the outputs and conclusions obtained should be used as new samples to be inputed to the system after they are verified so as to train the neural network. This will lead to further enhancement of the functional capabilities of the neural network.

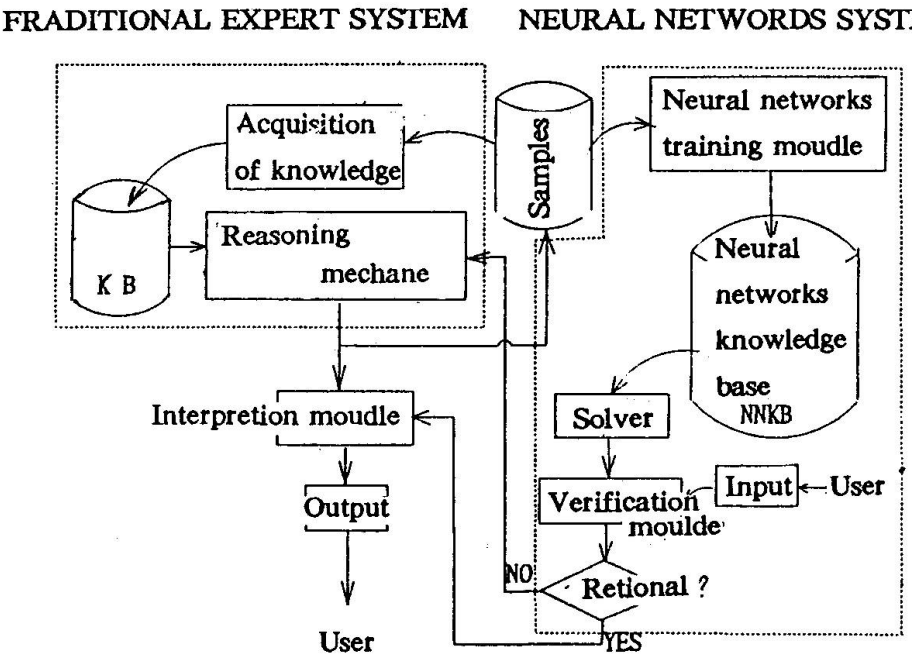


Fig. 6

The appolication of the neural network in engineering evaluation is just beginning. It has offered a new means to solve certain pending problems in our economic evaluation of railway construction projects.

REFERENCES

1. W. R. WIAN, Expert System for Roads. Monash University Press, 1988.

2. YIN HONGFENG, DAIRUWEI, Theory of Artificial Neural Networks Information Procoess. Pattern Recognition and Artificial Intelligence Vol. 3, No. 1, 1990.

3. ZHANG HONG BIN, Learning Algorithm of Artificial Neural Networks. Computer Science, No. 2 1990.

4. TANG YIQUN, Neural Networks and Neural Neworks Computer. Computer Magazine, Vol. 17,

No. 6, 1989.

5. B. IRIE, and S. MIYAKE, Capabilities of Three Layered perceptrons. Proc. of IEEE ICNN88, 1988.

6. S. S. RANGWALLA and D. A. DORNFELD, Learning and Optimization of Machining Operations Using Comtuting Abilities of Nearal Networks. IEEE Transctions on Systems, Man, and Cybernetics Vol. 19, No. 2, 1989.

7. YAO XIN, CHENGGOULIANG, Neural Networks Computer. Computer Engineering No. 8, 1990.

8. TANG YIQUN, Neural Networks and Neural Networks Computer. Computer Magazine Vol. 18, No. 1, 1990.

9. CHEN BAOLIN, The Theory and Algorithm of Optimization, QingHua University Press, 1989.

10. W. MYERS, The Combination in Expert System with Neural Networks Computer Science No. 5, 1991.

11. HONG JIARONG, Thinking Simulate Computer Science No. 4, 1991.

12. WANG IIANGZI XUYANG, Expert System for Economic Evaluation of Rail Way Construction Project. North Jiaotong University, 1990.

Representing Incomplete Knowledge with Assumptions

Représentation de connaissances incomplètes avec hypothèses

Darstellung unvollständigen Wissens mit Annahmen

Ian SMITH

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

S. BOULANGER

Research Assistant
Swiss Fed. Inst.of 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 Artificial Intelligence Lab.

Sylvie Boulanger obtained a B.Sc. in Civil Eng. from the Univ. of Alberta, Canada, and a M.S. from the Univ. of California, Berkeley, USA. Employed for over 5 years at CISC, she is now with the knowledge systems group at ICOM.

SUMMARY

This paper outlines a knowledge representation that accommodates two types of assumptions, defaul and preferences. Knowledge is categorised into three types, models-knowledge known before the ei gineering task begin, rules-knowledge which depends upon decisions made during completion of task, and control knowledge - knowledge related to task planning and conflict resolution, includin knowledge of when to introduce rule clusters, called knowledge islands. Assumptions are accomm dated within models and rules. When used with iterative reasoning strategies, such a representation ha much potential for supporting engineers during the preliminary stages of engineering tasks.

RÉSUMÉ

La représentation des connaissances de l'ingénieur peut être subdivisée en trois catégories: les "Modèles" en tant que personnification du savoir préalable à une tâche, les "Règles" en tant que capacité de décision au cours de l'exécution d'une tâche, et les "Connaissances des commandes" lors de la planification et de la maîtrise des conflits. Il faut y inclure la capacité de discerner le moment voulu pour appliquer les amas de règles (encore désignés par îlots de connaissances scientifiques). Les catégories de modèles et de règles de ce type de représentation s'appuient sur deux hypothèses, à savoir les normes techniques et les préférences personnelles. En liaison avec des stratégies itératives, cette forme de traitement des connaissances est à même de fournir un soutien énorme aux ingénieurs au cours de leurs travaux d'avant-projets.

ZUSAMMENFASSUNG

Die Ergänzung lückenhafter Information erfordert ein umfangreiches Ingenieurwissen. Zur Darstellung wird es in drei Kategorien eingeteilt: "Modelle" als Verkörperung der Vorkenntnisse, "Regeln" als Entscheidungswissen während der Bearbeitung einer Aufgabe, und "Steuerungswissen" bei der Planung und Konfliktbewältigung. Darin enthalten ist die Urteilskraft, wann Regel-Cluster (sog. Wissensinseln) einzusetzen sind. In die Kategorien der Modelle und Regeln sind zwei Typen von Annahmen einbezogen: Fachliche Standards und persönliche Präferenzen. Im Verein mit iterativen Strategien zum Ziehen von Schlüssen verspricht diese Form der Wissensaufbereitung eine beträchtliche Unterstützung der Ingenieurarbeit im Vorprojektstadium.

1 Introduction

There are very few tasks in civil engineering that are accomplished using only complete and exact information. Indeed, the ability to make a good decision in situations of incomplete knowledge is one of the most valuable skills of an experienced engineer. Consequently, knowledge systems which explicitly model strategies for accommodating incomplete knowledge have much potential for supporting civil-engineering tasks.

Engineers traditionally cope with inexact and incomplete information through use of statistical methods. More recently, methods such as default reasoning, fuzzy logic and neural networks have been proposed in order to provide additional strategies for treating imperfect information. These methods are most useful at different stages in the evolution of civil-engineering projects. Thus, they are not necessarily competing strategies; it is, however, important to employ methods that are most suited to the project stage under consideration.

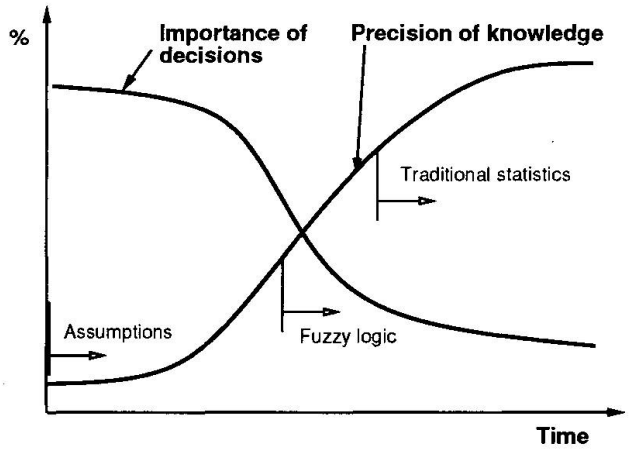


Figure 1: *The importance of design decisions and the precision of knowledge versus time.*

Two characteristic engineering relationships with time are described in Figure 1. The first relationship, *importance of decisions*, describes a trend which decreases slowly at first and then more rapidly until a minimum level is reached. This means that in any engineering task, the most important decisions are made at the beginning. Early decisions have the most influence upon factors such as costs, safety and environmental impact. The second relationship, *precision of knowledge*, describes an opposite trend. Precision (and accuracy) of knowledge is poor at first and increases as tasks near completion. Thus, Figure 1 illustrates a characteristic often used to define engineering practice; important decisions must be made in situations where knowledge is not reliable.

Also shown are positions on this curve where methods that treat imprecision become relevant for implementation. Traditional statistical methods typically require knowledge of all important parameters as well as the shape and size of their distribution. For example, codes recommend relationships derived from these methods. Fuzzy-logic methods (e.g. [1]) rely upon weight factors that usually represent the importance of parameters. Although less information than for traditional statistics is required to use these methods, information such as weight-factor distributions for each parameter is necessary. Therefore, such methods cannot be used at the beginning of engineering tasks. Various neural network methods (e.g. [2]) are proposed for use from the beginning of engineering tasks. However, their usefulness is dependent upon the quality of examples that were used to train the network. Furthermore, it is difficult to maintain performance when these methods are integrated within iterative and decision-making processes as the task proceeds.

The knowledge representation described in this paper results from a study that focuses on supporting decision making during the left portion of Figure 1. A framework for explicit representation of assumptions is presented, and it is shown how this framework can be integrated into a general framework for representation of knowledge to be used during preliminary stages of engineering tasks.

Also, an algorithm which accepts knowledge expressed according to this representation is described. Finally, an example in bridge design is used to illustrate an application in civil-engineering.

2 Representing assumptions

Assumptions are ubiquitous in civil engineering. Perhaps the only situation where civil engineers are able to perform tasks without making assumptions is found during their education, when introductory textbook exercises are completed and simple laboratory experiments are performed. Two types of assumptions are proposed, default and preference. Default information includes all information that is known to be "usually" the case. A great deal of common sense knowledge is default information under this definition. For example, the following statements :

- "Buildings normally have four exterior walls"
- "In this region, piles are not needed for low rise office buildings"
- "Bridges usually have sidewalks on both sides"

contain default information. This type of information is used to make decisions until more information is known.

The second type of assumption, a preference, includes all desired decisions. Therefore, this information reflects a wish to proceed a certain way. If too many problems arise from adopting a preference assumption, the assumption is weakened. Examples of preference assumptions are :

- "Given a choice, study the cheapest alternative"
- "If slope is unstable then avoid placement of bridge piers on the slope"
- "If there is a high risk of earthquakes then continuous bridge spans is a good design for span types"

There are two important differences between defaults and preferences. Firstly, defaults reflect a rough statistical understanding of previous experience in the field. Such an understanding may have no relation to any basic principle within the domain. Preferences, however, reflect the existence of knowledge which is hard to represent more explicitly. Thus, a preference has some relationship to a basic principle which influences the domain. Under this definition, a great deal of civil engineering knowledge can be expressed in preferences since many decisions are influenced by factors that are difficult to model. Such factors include environmental considerations, politics, economic conditions and aesthetics.

The second difference between defaults and preferences is found in their behaviour during situations of conflict with new information. When engineers encounter specific information that conflicts with a default assumption, they simply drop this assumption. Since a default assumption is based solely on empirical observation, there is no reason to do otherwise. However, when a preference assumption is in conflict with more relevant information, it is not dropped completely. Engineers "weaken" the assumption in order to accommodate other information and remember it later if subsequent inferencing creates a situation where it can be re-established.

3 Representing knowledge at preliminary stages of engineering tasks

This section describes a framework for representing defaults and preferences within two knowledge structures, models and rules. Models represent information that is available independent of the specific task. Generally, models are based on physical principles, domain theories and contextual parameters. For example, a geometric layout of building elements is a model reflecting physical principles of connectivity; a structural analysis program is based upon models of stress analysis; and a building

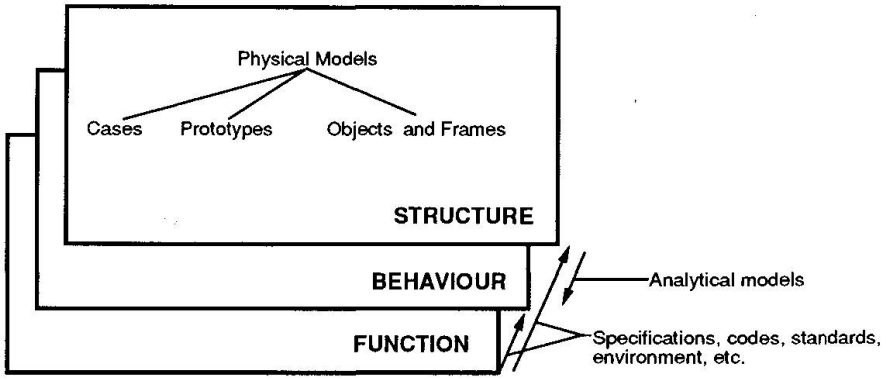


Figure 2: *Types of models along with their planes of abstraction*

code is a model of physical principles and context, such as local requirements and local environmental factors such as wind, earthquake and snow loads.

Models can be expressed on three levels of abstraction - structure, behaviour and function, see Figure 2. Physical models contain information principally on the structural level although important information can exist at all levels [3, 4]. Physical models may contain a great deal of specific information, such as in cases, or relatively little, such as in fundamental hierarchies of elements in a frame system.

Models that contain information that is orthogonal to the structure, behaviour, function planes include analytical models and models of context and user requirements such as building codes, specifications and standards. These models facilitate transformation from one plane to another in order to perform reasoning at the appropriate abstraction. For example, once the structure is defined and loads are known, analytical models are able to transform geometric information in the structural abstraction into stresses and deformations in the behavioural abstraction. Building codes help transform functional requirements into behavioural needs such as maximum deflections and ultimate stresses. These two transformations enable a standard design evaluation to be performed at the behaviour level. In addition to a function-behaviour transformation, codes and specifications include requirements on the geometry of elements and spaces, thereby influencing structural abstractions.

Physical models provide good opportunities for representing general default assumptions. These assumptions can be contained in slots within representations of prototypes and frames. Since cases *already* contain values of parameters, these values act as default values for new contexts where the case is inserted and therefore, default values do not require a special representation.

The second knowledge structure is rules, see Figure 3. Rules contain information that is dependent on engineering decisions made during execution of the task. Rules are divided into two categories, fixed and assumptions. Fixed rules are rules that must be satisfied in order to complete the task. Fixed rules are further divided into two types according to the knowledge they represent - physical principles and technological considerations. Rules related to physical principles represent knowledge about requirements such as geometry - for example, the sum of all bridge spans must equal the length of the bridge. Rules based on technological considerations exist in order to avoid solutions that are technologically infeasible - for example, during construction, helicopter erection is possible only for elements whose weight do not exceed the maximum permitted by the helicopters available to contractors.

Assumptions are the second category of rules. Here lies the knowledge which is most important for effective support of preliminary stages of engineering tasks. As explained in the previous section, two types of assumptions are distinguished - default and preference. Default rules provide default values that are more specific to certain situations than those defaults supplied by physical models. Preference rules introduce knowledge which is difficult to model explicitly. Such knowledge is most easily represented using a rule structure in order to provide the opportunity to identify specific

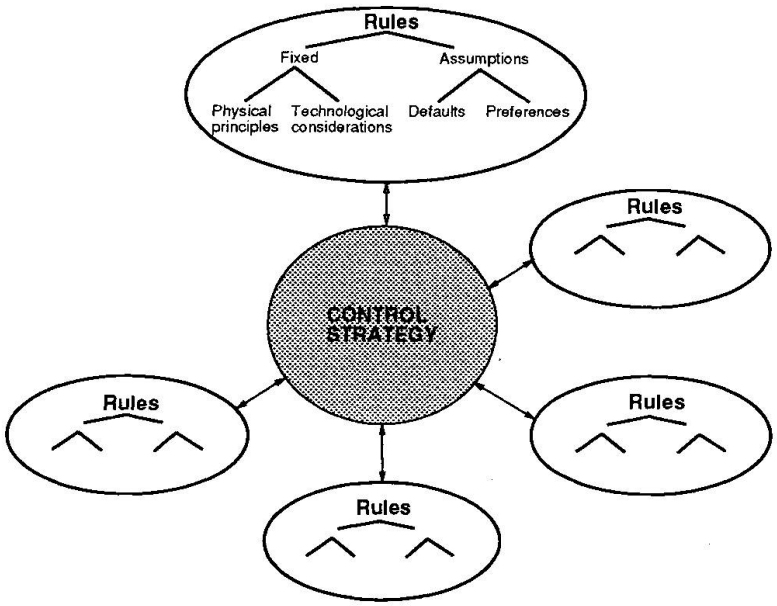


Figure 3: *Rule classifications organized into groups connected to a control strategy*

conditions (in the "if" portion of the rule) for consideration of the preference. An example of the use of assumption rules is given later in the paper.

The ellipses in Figure 3 represent collections of knowledge into groups where all knowledge within an group is consistent. A single domain may have many groups. We believe that such an organization of knowledge is essential to developing a knowledge representation on a large scale. Systems that require creation of one large and consistent knowledge base impose impossible requirements upon developers of the knowledge base and this is only amplified with time, as new information is added and existing information is changed.

The circle labeled control strategy in the center of Figure 3 has two roles. Firstly, it manages conflict resolution when information from different groups is contradictory. The second role determines when each group is added to memory. Since the system resolves conflict through backtracking, relaxing preferences and undoing the effects of default assumptions, the order of appearance of conflicts affects the result of the system. Note that this approach is not a blackboard-type strategy. More details are given in the next section.

4 Reasoning with assumptions

The algorithm employed to control the introduction of rules and the resolution of conflicts is illustrated in Figure 4. Since the focus of this paper is on knowledge representation, the description of this algorithm is brief. More details are given in [5]. Initial conditions related to the task are taken interactively from the user or from a file. These conditions may be used to organize knowledge groups into levels of importance or alternately, the user may select an importance hierarchy from a case base of task plans. These hierarchies are discussed further at the end of this section.

Rules in the lowest level are first added into the rule engine. If the initial conditions do not contain enough information to fire a given rule, default values are taken from the physical model. When information is not explicitly contained in the physical model, analytical models may be employed to derive it. The information inferred by these rules often take the form of constraints which are propagated through one another in order to fix consistent constraints. They are then checked for feasibility since algorithms for propagating constraints on continuous variables may not result in a feasible solution. Considering rules only at the first level, there must be consistent constraints resulting in a feasible solution (by definition, since each knowledge group must contain consistent knowledge) so the feasibility test passes and the next level of rules is loaded into the rule engine.

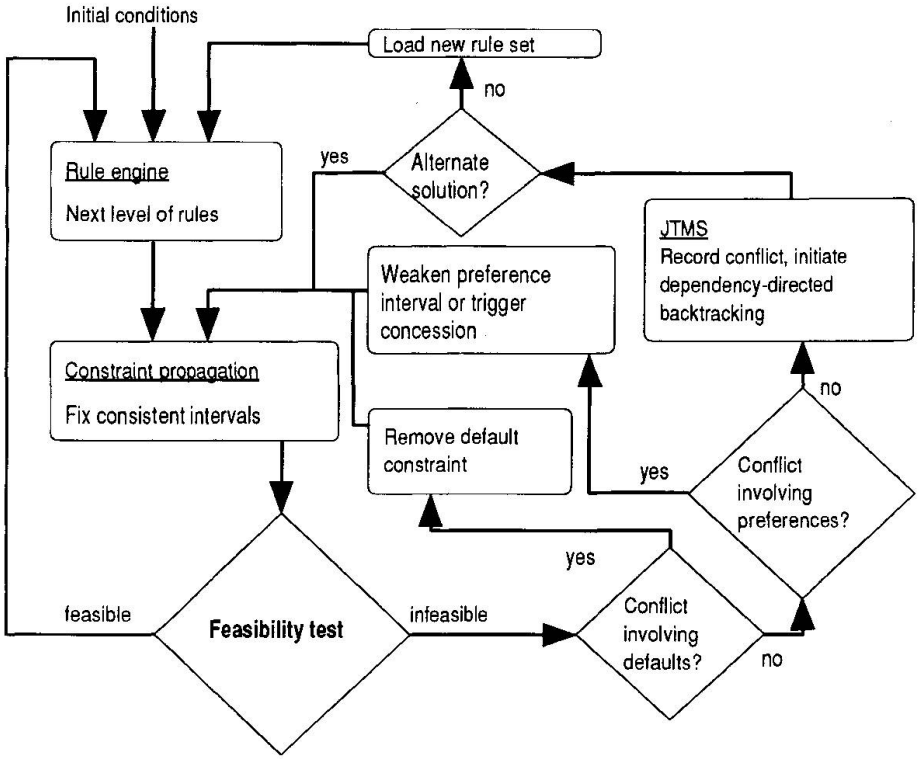


Figure 4: *The algorithm used to reason with defaults and preferences.*

Once the second rule level is processed, infeasible solutions become possible. The system resolves conflicts in the following manner. Justifications for all constraints are recorded. If the conflict involves defaults, the default information is dropped, inferred knowledge is retracted, constraints are propagated again and these are rechecked for consistency. If more than one piece of default information is involved in the conflict, defaults originating from the physical model are dropped first, followed by defaults in the lowest rule level up to the highest.

If a conflict remains after all default information has been dropped, it is examined for the presence of preference rules in the justification list. Here preference rules are weakened in a manner which is not the same as dropping defaults; weakened rules influence valid intervals and they are kept in memory in case a situation arises where they can be fully reinstated. If more than one preference is involved in the conflict, preferences on the lowest levels are weakened first. Since such weakening can result in a drastic change in the solution space, some preferences include concession clauses. In the case of a conflict, concession clauses are used to retract the preference and substitute it with a new rule in order to avoid forcing the system to backtrack to the position where the first preference was inferred. Rules in concession clauses can be of two types, preference or fixed. If preference type concession clauses do not remove the conflict, weakening occurs normally as described above. When the concession clause contains a fixed rule, subsequent conflicts are treated the same way as with fixed rules. This is described next.

When conflicts remain after all preferences have been weakened, the system performs dependency-directed backtracking in order to investigate alternative solution paths. This strategy, along with those that analyse justifications for conflicts, employs a truth maintenance system [6], adapted for intervals of continuous variables. Here the closest available alternate path is taken and often, the system does not require significant backtracking. If an alternate solution path is available, that path is taken and constraint propagation is carried out on the new solution.

If no path is available, a new rule set, consisting of another collection of knowledge groups, is loaded. Rule sets are created for each solution type. For example, different rule sets are needed for different types of bridge designs such as cable stayed, arch, suspension, beam and truss bridges. The

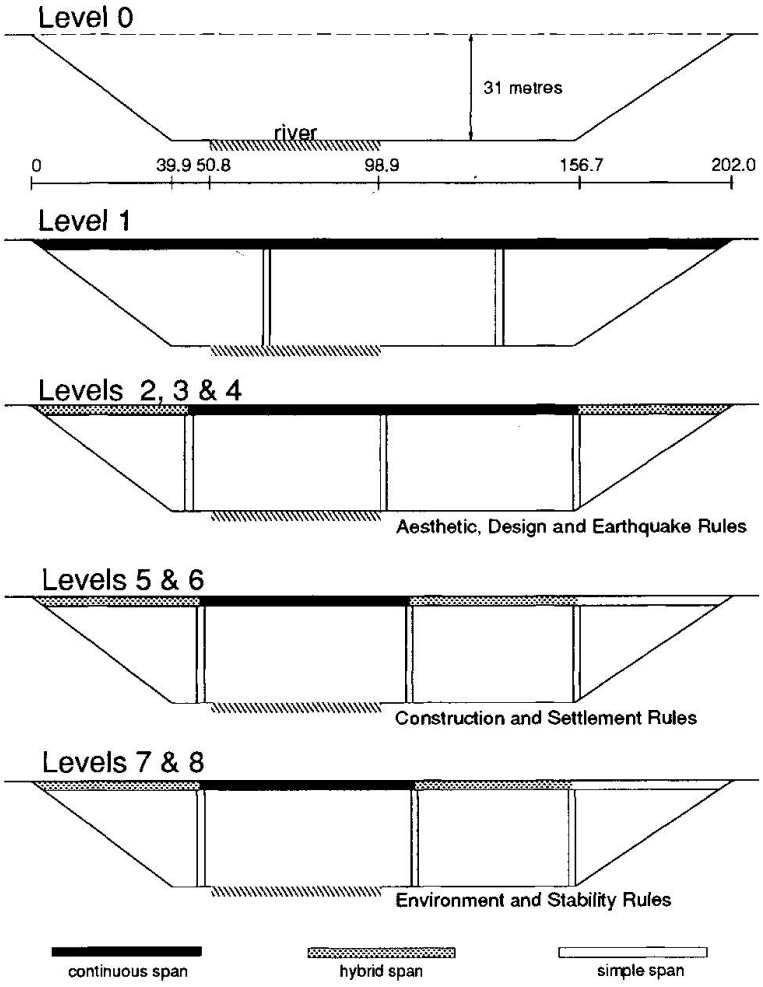


Figure 5: *Evolution of the design of the French Creek Bridge.*

order of priority given to such sets is either predetermined or set through inferences performed on parameters on initial conditions.

Knowledge groups can be organized according to different types of knowledge that influence the task. For example, individual knowledge groups in a bridge design system can refer to design criteria such as aesthetic requirements, construction feasibility, costs, safety, long-term serviceability, detail design and traffic requirements. The order of introduction into the system establishes the relevant importance of each criterion. In this way, the engineer influences how conflict resolution is performed and ultimately, the final solution proposed by the system. Through investigating different hierarchies for each knowledge group, engineers are able to evaluate various solutions which are all consistent with the knowledge contained in the system. Therefore, this knowledge representation, along with the reasoning strategy described in this section, is capable of reflecting the fact that engineers find different solutions for the same problem, depending upon the importance they place on relevant criteria.

5 An example in design

The knowledge architecture described in the previous sections has been implemented in a system called PRELIM, a system for the preliminary design of civil-engineering structures. This example is drawn from a recent bridge project, the French Creek site in British Columbia, Canada. The bridge site is a 202 metre long valley with a 48 metre wide river situated in the second quarter of the valley. The location is remote, access is difficult and the site is situated in a zone of high seismic risk. Furthermore, the construction schedule overlaps with an important reproductive period of the fish population. This

LEVEL	*SPAN-1*	*SPAN-2*	*SPAN-3*	*SPAN-4*
Level 1 - Start-Up				*note :*
span intervals :	[67.3 67.3]	[67.4 67.4]	[67.3 67.3]	*span values*
span type :	continuous	continuous	continuous	*are in metres*
Level 2 - Aesthetics				
span intervals :	[50.5 50.5]	[50.5 50.5]	[50.5 50.5]	[50.5 50.5]
span type :	continuous	continuous	continuous	continuous
Level 3 - Design				
span intervals :	[48.0 53.0]	[48.0 53.0]	[48.0 53.0]	[48.0 53.0]
span type :	continuous	continuous	continuous	continuous
Level 4 - Earthquake				
span intervals :	[42.6 47.1]	[53.3 58.9]	[53.3 58.9]	[42.6 47.1]
span type :	hybrid	continuous	continuous	hybrid
Level 5 - Construction				
span intervals :	[42.6 47.1]	[53.3 58.9]	[15.0 50.0]	[42.6 47.1]
span type :	hybrid	continuous	continuous	hybrid
Level 6 - Settlement				
span intervals :	[45.8 50.7]	[57.3 63.3]	[45.8 50.0]	[43.0 47.5]
span type :	hybrid	continuous	hybrid	simple
Level 7 - Environment				
span intervals :	[45.8 50.7]	[59.2 63.3]	[45.8 49.1]	[43.0 47.5]
span type :	hybrid	continuous	hybrid	simple
Level 8 - Stability				
span intervals :	[45.8 50.7]	[58.2 63.3]	[45.8 46.8]	[45.3 47.5]
span type :	hybrid	continuous	hybrid	simple
Final Values	48.2	61.7	46.8	45.3
Real Values	47.0	64.5	47.5	43.0

Underlined numbers indicate results after overriding, up-dating, weakening or re-activating a rule

Figure 6: *Modification of span lengths as rule levels are introduced.*

means that no piers or temporary props can be erected in the river and to the right of the river. Finally, the soil conditions are poor with slopes on each side of the valley approaching instability.

The evolution of the design is shown schematically in Figure 5. As the system processes rules from rule level 0 to level 8, parameters such as number of spans, pier position and span type are modified. The evolution of span lengths with rule level is given in Figure 6. A partial list of rules is provided in Figure 7. The physical model for this example is expressed in terms of a frame system. The frame corresponding to this bridge type contains, amongst other information, the following default instances:

- no. of spans : 3
- span type : continuous
- span restrict : 15 to 75 meters

The site configuration is shown at level 0 on Figure 5. Level 0 contains general rules that reflect physical principles such as geometric consistency. Since information about the bridge emerges as the design progresses, PRELIM uses default rules to get started. When level 1 is reached, the "default" or start-up design has 3 continuous spans of equal length. The size and location of the river zone is not yet considered. At level 2, an aesthetics-type rule determines an appropriate number of spans for the height of the valley using aspect ratios and the golden rectangle [7] and resulting in a choice of 4 spans. The rule on level 2 is a preference rule and therefore, the values given by the default rule on level 1 are dropped. Design rules at level 3 assign span factors for each span according to the type of span. Using these factors in a ratio with the total length of the bridge, as well as a rule which allows a certain variation from this factor, preferred intervals for spans are developed, see Figure 6. Since preference rules at level 3 are considered to be more important than preference rules at level 2, the values fixed for spans correspond to the values calculated by the rules at level 3. However, the constraints corresponding to the rule at level 2 are weakened, not dropped. Its effect is reinstated if further inferencing weakens the effect of the rules on Level 3.

CONSIDERATION	TYPE	LEVEL	RULE
General	Fixed	0	If number of spans = n then sum of the spans (1 to n) = L
General	Fixed	0	If span type i is simple and span type i-2 is continuous then span type i-1 is hybrid
General	Fixed	0	If number of spans = n then number of piers = n - 1 and span i+1 = pier i+1 - pier i
Start-Up	Default	1	If number of spans = n then span i = L/n
Aesthetic	Preference	2	If H/L is between 1/15 and 3/5 and H is between 15 and 75 then typical span varies from 1.5 to 1.75 H and number of spans = L / typical span
Design	Preference	3	If beam type is constant depth and span type i is continuous then span factor i is 10
Design	Preference	3	If beam type is constant depth and span type is hybrid then span factor i is 8
Design	Preference	3	If beam type is constant depth and span type is simple then span factor i is 7.5
Design	Preference	3	If span i has factor i then span i is (factor i/ sum of the factors) x L ± 5%
Earthquake	Preference	4	If earthquake region is high then interior span type is continuous and exterior span type is hybrid
Construction	Preference	5	If construction method is launching and beam type is constant depth then span i (all spans) < 50
Construction	Preference Concession	5	If construction method is launching and beam type is constant depth then a maximum of one span can be more than 50 and less than 65
Settlement	Preference	6	If soil condition is very poor then all span types are simple
Settlement	Fixed Concession	6	If soil condition is very poor and river situation in valley is left then right exterior span type is simple
Environment	Fixed	7	If pier is at edge of sensitive river zone then move pier out of zone
Stability	Preference	8	If slope is unstable then move pier off slope

span : span value in metres pier : pier position in metres

Figure 7: *A partial list of rules employed.*

At level 4, earthquake requirements change the continuity of the spans - two simple supports at the abutments change the exterior spans to a hybrid span type (simple support at one end and continuous at the other). The system then modifies the span factors for each span in order to reflect these new continuity conditions. This means that the span factors are updated to 8-10-10-8. Construction requirements at level 5 result in the elimination of construction by crane considering soil conditions and access difficulties. Since crane erection is a default instance in the frame, PRELIM now searches for another method. At this point in the development, the system selects the next method of construction from the list of instances in the frame hierarchy. The next construction method is launching. The level 5 rule for launching is a preference limiting the spans to 50 metres. However, a conflict with a physical principles rule at level 0 occurs; the sum of the spans is smaller than the length of the valley. Before weakening what has been inferred thus far, PRELIM looks for a concession clause attached to the most recently added rule which is involved in the conflict. A concession rule is found that accepts one span between 50 and 65 metres. Span 2 is then relaxed to a greater value than 50 and span 3 is limited to 50 after weakening the corresponding rule from level 3.

Settlement considerations in level 6 initially fire a preference for simple supports in order to reduce the stress at the abutments. However, this creates a conflict with earthquake requirements for continuity on level 4. PRELIM finds a concession rule which accepts one simply supported exterior span, the one farthest from the river. Hence span 4 is now simply supported. From a physical principles rule at level 0, span 3 becomes hybrid. Span factors are updated and the design rules from level 3 are re-activated from a weakened state, giving five percent intervals around proportions of 8-10-8-7.5. At this point, construction criteria are still respected but some earthquake criteria are weakened. This demonstrates the persistence of preference rules. Rather than having their effect eliminated entirely,

as is the case for default rules, they remain in memory and are re-established when justifications for weakening disappear.

At levels 7 and 8, PRELIM loads rules from groups that focus upon pier position. At level 7, an environmental concern for spawning beds prevents a pier positioning in the river and 11 metres to the right of the river. Although no direct conflicts occur, a level 0 rule relating spans as a function of pier positions up-dates and narrows the range of possible values for spans 2 and 3. At level 8, there is a strong preference to avoid placing a pier on an unstable slope. As was the case for level 7, the ranges of possible values, this time for spans 3 and 4, are modified.

Final values of the spans are selected by applying the design rules similar to those on level 3 (without the five percent tolerance rule), and are adjusted with fixed rules of physical principles. The values obtained compare well with the values adopted for the the project in reality. Therefore, using PRELIM, a rough simplification of the actual preliminary design process results in values that are close to those for the as-built bridge.

6 Conclusions

Engineers often make important decisions when information is not reliable. Therefore, assumptions are an essential part of knowledge employed during preliminary stages of engineering tasks. When this knowledge is divided into defaults and preferences and then integrated into models and rules organized into knowledge groups, there is much potential for representing assumptions in a flexible manner. With such a representation, iterative reasoning strategies use priorities attached to knowledge groups to resolve conflicts rationally and simulate the evolution of decisions as more information becomes available. Preliminary engineering tasks can thus be supported in situations where knowledge is incomplete and contradictory.

Acknowledgements

Funding for this research was provided by the Swiss National Science Foundation. The authors would like to thank Boi Faltings for useful discussions and Djamila Haroud for development and implementation of the reasoning algorithm. Also, the contribution of Dr. Peter Buckland, Buckland and Taylor Consulting Engineers, North Vancouver with respect to the knowledge used for the design case described in this paper is gratefully recognized.

References

[1] **WOOD, K.L and ANTONSSON, E.K.** "Computation with imprecise parameters in engineering design: background and theory" J. of Mechanisms, Transmission and Automation in Design, 111, 1989.

[2] **GARRETT, J.H. et al** "Engineering applications of neural networks" Journal of Intelligent Manufacturing, 1992.

[3] **GERO, J.** "Design prototypes: A knowledge representation schema for design", AI Magazine, Vol 11, No 4, 1990.

[4] **UMEDA, Y. et al** "Function, behaviour and structure" Applications of Artificial Intelligence in Engineering V, Springer-Verlag, 1990.

[5] **FALTINGS, B., HAROUD, D. and SMITH, I.** "Dynamic constraint propagation with continuous variables" 10th European Conference on AI, ECAI'92, Vienna, 1992.

[6] **DOYLE, J.** "A truth maintenance system" Artificial Intelligence, 12, 1979.

[7] **LEONHART, F.** Bridges, aesthetics and design, Verlags-Anstalt, Stuttgart, 1982.

Reconstructing Conceptual Models of Structure

Reconstruction des modèles de conception des structures

Zur Rekonstruktion konzeptioneller Modelle aus Bauwerken

J.T. DE GELDER
TNO
Delft, The Netherlands

L.F.M. VAN GORP
TNO
Delft, The Netherlands

G.L. LUCARDIE
TNO
Delft, The Netherlands

SUMMARY

Until now, conceptualization methods applied in the modelling of structures are based on the assumption that, in essence, all attributes for creating a concept are provided initially. Furthermore, it is assumed that the classifications of these attributes are a priori fixed and possess an unconditional status. Recent advances in Artificial Intelligence are stating that relevant descriptive attributes are not necessarily given a priori but should be acquired by reasoning about the goals of classification. A goal-oriented approach initiates a process wherein conditional attributes and dynamic classifications are reconstructed. This paper shows that a conceptual model of a steel structure should take into account the goals of different parties involved in the building and construction industry.

RÉSUMÉ

Jusqu'à nos jours, les méthodes de conception appliquées dans la modélisation des structures étaient fondées sur l'hypothèse que tous les attributs d'élaboration d'un concept étaient déjà fournis au départ et, en outre, que leur classification existait inconditionnellement. Les récents progrès de l'intelligence artificielle montrent toutefois que les attributs descriptifs remarquables ne sont pas donnés à priori, mais que le raisonnement devrait permettre de les déduire des objectifs de classification. Une approche à orientation objective donne naissance à un processus de reconstruction d'attributs conditionnels et de classification dynamique. En s'appuyant sur un exemple de structure métallique, l'article montre comment le modèle associe les objectifs différentiels des intervenants au projet et à la construction.

ZUSAMMENFASSUNG

Ueblicherweise wird bei der Abstraktion von Bauten zum dahinterstehenden Entwurfskonzept unterstellt, dass alle Attribute des Konzepts von Anfang an gegeben sind und ihre Klassifikation bedingungslos feststeht. Jüngere Fortschritte der künstlichen Intelligenz zeigen jedoch, dass dies nicht der Fall sein muss, sondern relevante beschreibende Attribute erst aus dem Nachdenken über die Klassifizierungsziele abgeleitet werden sollten. Durch den zielorientierten Ansatz entsteht ein Prozess zur Rekonstruktion bedingter Attribute und einer dynamischen Klassifikation. Am Beispiel eines Stahlbaus wird gezeigt, wie das konzeptionelle Modell die unterschiedlichen Zielsetzungen der am Entwurf und Bau Beteiligten einbezieht.

1. INTRODUCTION

Conceptual modelling is an essential activity in the development of knowledge based systems. In current practice, conceptual modelling of structures often leads to the development of generic models. Generic models, however, have the problem, that they are not really functional for a specific application. Generic models are the consequence of following a conceptual modelling theory, which we call the theory of extensional classifications. In this paper the limitations of this theory are described and an alternative theory, known as the theory of functional classifications, is proposed. In chapter 2 these two theories on conceptual modelling are described. By giving an example of the modelling of steel structures in chapter 3, it is shown that the theory of functional classifications, is preferable to the theory of extensional classifications. Finally, chapter 4 discusses some advantages of using the theory of functional classifications in conceptual modelling.

2. THEORY

2.1 Knowledge level

In his presidential address for the American Association of Artificial Intelligence (AAAI), Newell introduced a new computer system level: the knowledge level [10]. At this conceptual level knowledge about an application-area is defined [1,2,10]. Modelling at the knowledge level permits the development of implementation independent conceptual models, which better reflect the reality of an application area [2,8,15].

2.2 Conceptual modelling

Conceptual models consist of concepts. The meaning triangle of Ogden en Richards, cited in [14] explains what a concept is (figure 1). A concept refers to a physical object and is symbolized by a symbol. The fact that we symbolize the object which we use to take a seat as "a chair", is because we have an idea of what a chair is. This idea is called a concept.

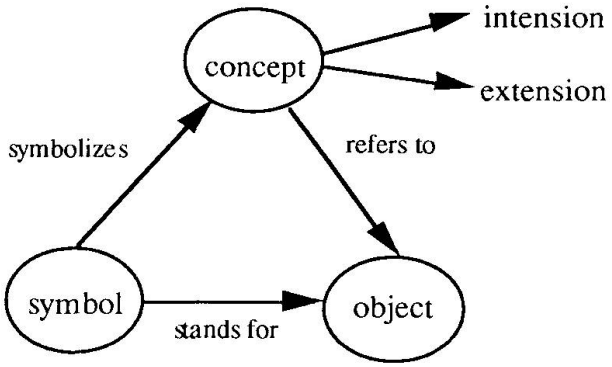


Fig. 1 Meaning triangle

Conceptualization involves reconstruction of the meaning of concepts. Each concept has an intension, also called an object-type, and an extension. The intension of a concept is a set of requirements, which should be met by an object in order to belong to the class covered by the concept. The intension of the concept "chair", for instance, could be the existence of a seat, a back and at least one leg. The extension of a concept is a set of objects, satisfying the intension. So, in the example, all objects having a seat, a back and at least one leg are classified as chairs.

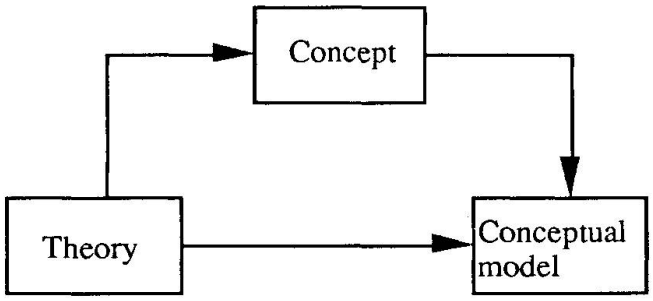


Fig. 2 Conceptual modelling using a theory

For reconstructing the meaning of concepts a theory about concepts is required, that prescribes how the meaning of concepts should be reconstructed (figure 2). There are several theories dealing with the reconstruction of concepts. Two of them are of special interest and will be described below.

2.3 Theory of extensional classifications

In practice the theory of extensional classifications is often applied, in many cases implicitly. In this theory a conceptual model is developed by analysing the objects in reality which have to be modelled [3,18]. For instance, a conceptual model for a chair is developed by analysing the common properties of chairs. In fact the intension of a concept is reconstructed by analysing extensions of the concept.

2.4 Theory of functional classifications

2.4.1 Goal-oriented

In contrast with the theory of extensional classifications, the theory of functional classifications [5,11,12,13], reconstructs the intension of a concept by following a goal-oriented approach. The theory of functional classifications states that the decision on which attributes and constraints to reconstruct for the intension of a concept totally depends on the goal of classification. Having a goal or a function, one has to find out which knowledge is required to reach that goal or to perform that function.

2.4.2 Functional equivalent object-types

In reconstructing the intension of a concept by following a goal-oriented approach functional equivalent object-types are reconstructed. In reality there are often several ways to reach a certain goal or to perform a certain function. If a concept is symbolized by "chair" and the only requirement is that someone should be able to take a seat, a table meets this requirement too. So, a table is functional equivalent to a chair if the function is "taking a seat". In the theory of extensional classifications a table and a chair would have been modelled differently, because different properties are recognized. But, applying the theory of functional classifications most of these properties won't be modeled, since they are irrelevant to perform the function "taking a seat".

2.4.3 Context dependency

Another essential aspect in the theory of functional classification is the relevance of a context. The reconstruction of concepts is also directed by a context. The meaning of the concept "good football player", for instance, totally depends on the context in which this concept has to be reconstructed. In the context of a local club team this concept has a completely different meaning than in the context of a world class team. More precisely, a good football player in a local club team has completely different properties than a good football player in a world class team. This aspect is illustrated by figure 3.

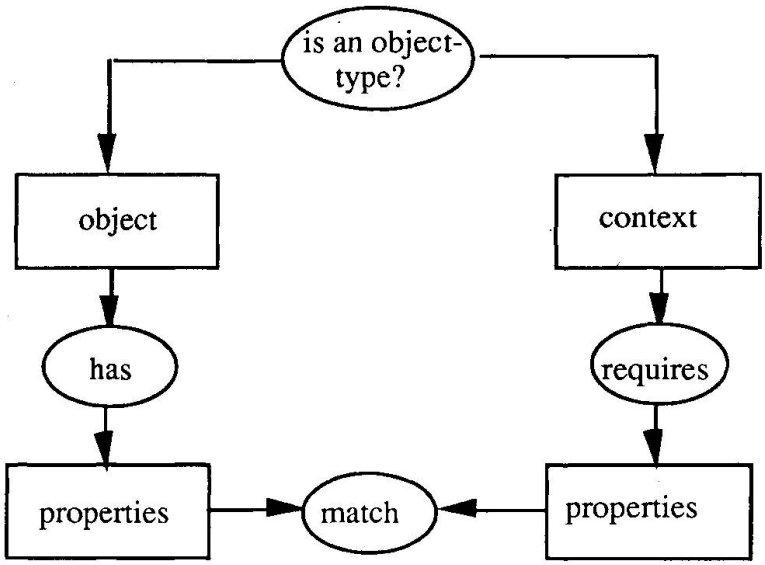


Fig. 3 Context dependency in conceptual modelling

A specific context requires specific properties. An object defined by attributes and constraints is an object-type of the conceptual model if and only if the properties of the object-type defined by attributes and constraints match the properties required by the context.

For the reconstruction of a conceptual model formal techniques are essential. In conceptual modelling work performed by using the theory of extensional classification, often a data modelling technique, like NIAM [18] or IDEF1x [3] is applied. These techniques are not goal-oriented and don't support the reconstruction of functional equivalent object-types. For this reason data modelling techniques are not very useful, if conceptual modelling is performed by using the theory of functional classifications. In the next chapter decision table techniques are applied for conceptual modelling of steel structures using the theory of functional classifications.

3. CONCEPTUAL MODELLING OF STEEL STRUCTURES

3.1 Realisation of steel structures

The process of realizing steel structures can be split into three sub processes, design, detailing and manufacturing.

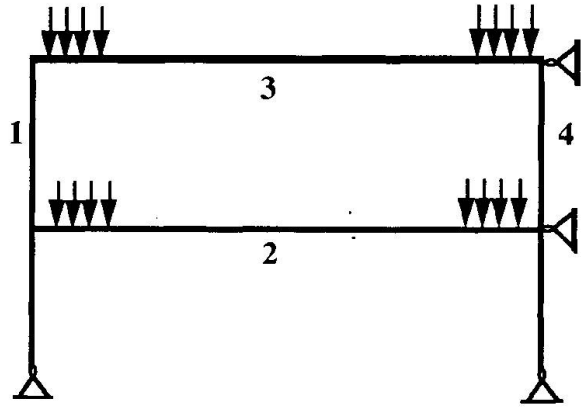


Fig. 4 Portal structure with loads

Starting point in the design process is the design of the topology. Figure 4 displays the topology of a portal structure. The portal structure consists of two columns and two beams.

The topology of the portal structure and the loads are the basis for designing the profiles and calculating the required capacity of the connections. Figure 5 displays the portal structure again at the end of the design process, which is the start of the detailing process. At this moment the required capacities of the connections are known.

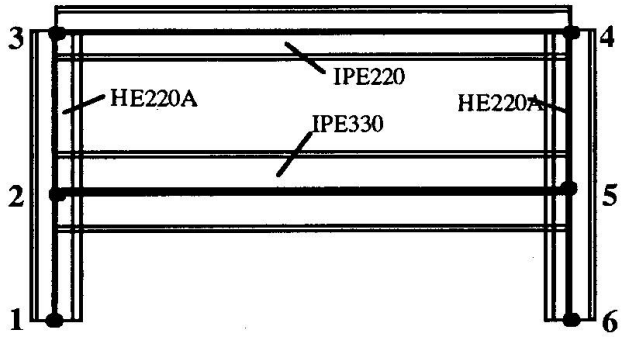


Fig. 5 Portal structure with profiles

In the example only the moment capacity, which is an indication of the strength of the connection, is considered. In the detailing process connections have to be designed, which meet the required moment capacity.

The example concentrates on node 2 and an end plate connection is detailed for it. In column to beam end plate connections, the actual joint between the column and the beam is provided by an end plate, that is welded to the beam and bolted to the column. It is calculated that the required moment capacity in node 2 is 60 kNm.

3.2 Modelling steel structures using the theory of functional classifications

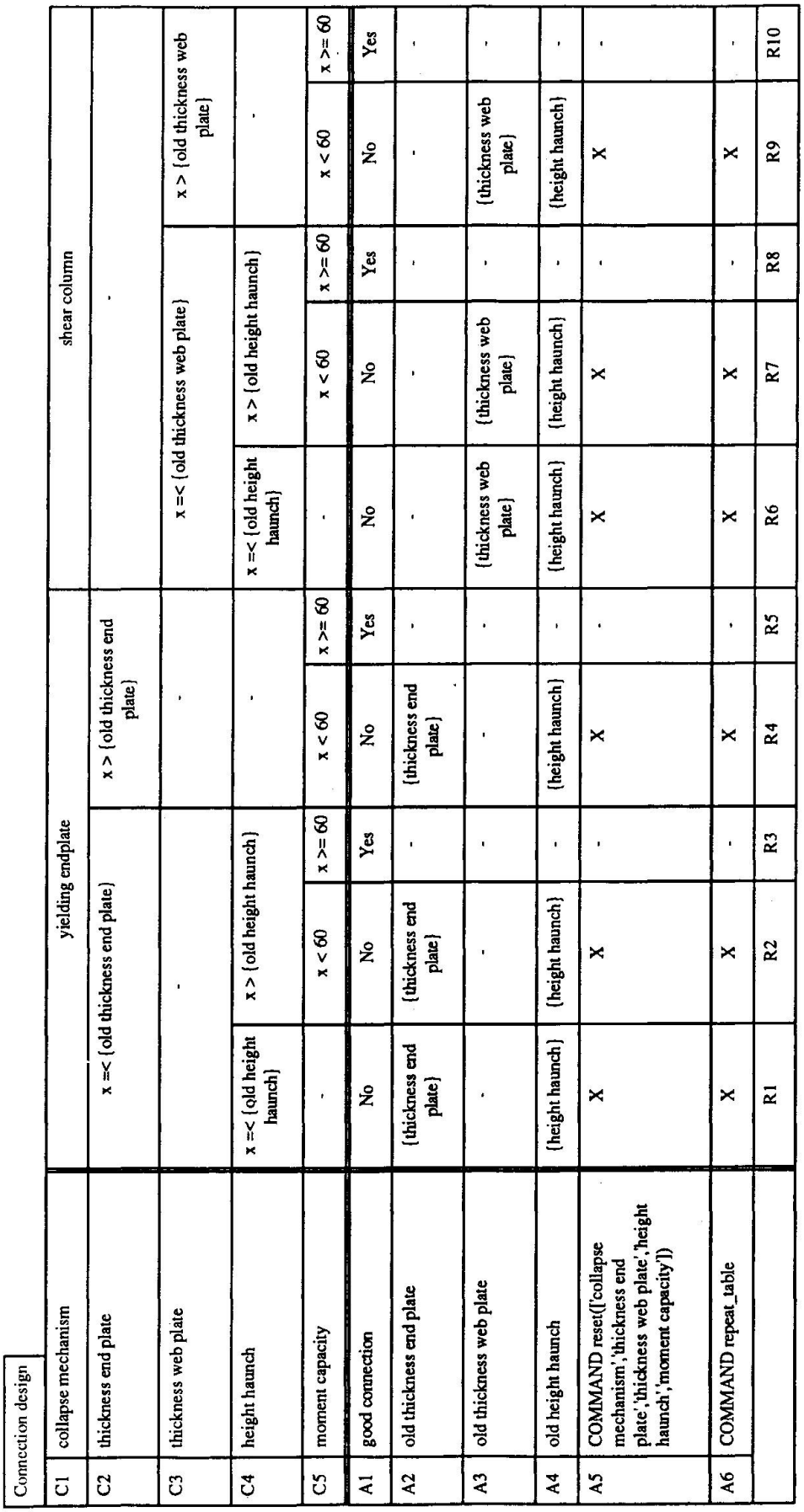
The process to detail the connection at node 2 is displayed by the decision table in figure 6. The decision table is made using DTSS (Decision Table System Shell), an interactive graphic tool to define and consult knowledge in a decision table [7].

A decision table consists of two parts, a condition part and an action part. The condition part defines the conditions in which a specific action has to be performed.

The decision table technique supports a goal-oriented modelling approach. In figure 6 the modelling goal is defined by action A1 "good connection". To detail a good connection 5 conditions are evaluated. In reality more conditions are relevant, like, for instance, bolts and welds, but these are ignored in the example.

The decision table shows that a connection is good only if the condition moment capacity is greater than or equal to 60. The moment capacity is the result of a calculation done by a connection analysis program. So, after each detailing step the moment capacity is calculated and compared to 60. If the calculated value is less than 60, the actions A2 to A6 are performed. The actions A2 to A5 set values for the next detailing step. Action A6 actually activates the next detailing step. If the calculated moment capacity is greater than or equal to 60 only action A1 is performed and the detailing process is finished.

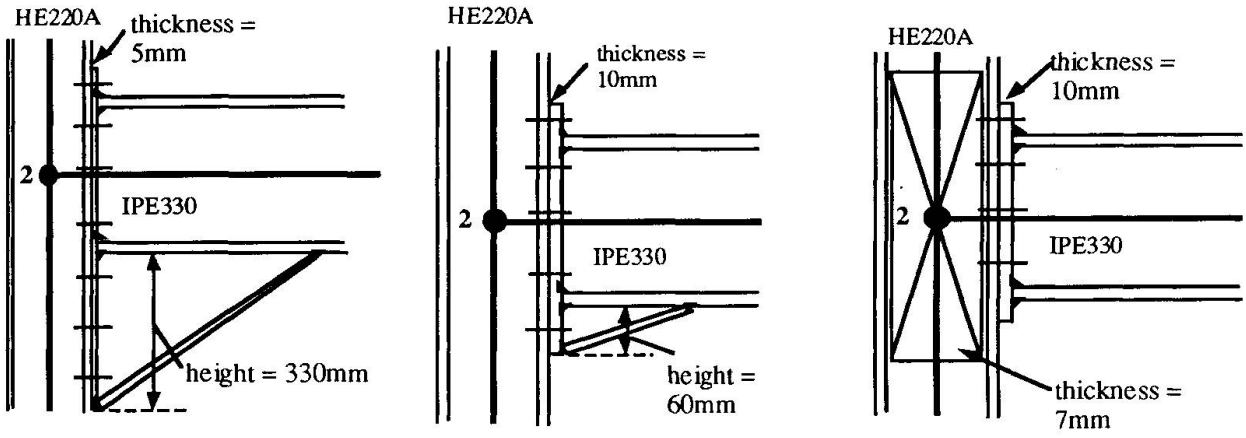
If the moment capacity is less than 60, the detailed connection is not good yet and has to be strengthened by increasing the dimensions of elements or including new elements, like a haunch or a web plate [17]. The decision on which dimensions to increase or which new elements to include depends on the determining collapse mechanism of the connection (condition C1). Like the moment capacity, the determining collapse mechanism is also the result of a calculation done by a connection analysis program. In the example only the collapse mechanisms "yielding of the end plate" and "shear of the column web" are considered.



Connection design																
C1	collapse mechanism	yielding endplate						shear column								
C2	thickness end plate	x =< {old thickness end plate}			x > {old thickness end plate}			-								
C3	thickness web plate	-			-			x =< {old thickness web plate}			x > {old thickness web plate}					
C4	height haunch	x =< {old height haunch}	x > {old height haunch}		-			x =< {old height haunch}	x > {old height haunch}		-					
C5	moment capacity	-	x < 60	x >= 60	x < 60	x >= 60		-	x < 60	x >= 60	x < 60	x >= 60				
A1	good connection	No	No	Yes	No	Yes		No	No	Yes	No	Yes				
A2	old thickness end plate	{thickness end plate}	{thickness end plate}	-	{thickness end plate}	-		-	-	-	-	-				
A3	old thickness web plate	-	-	-	-	-		{thickness web plate}	{thickness web plate}	-	{thickness web plate}	-				
A4	old height haunch	{height haunch}	{height haunch}	-	{height haunch}	-		{height haunch}	{height haunch}	-	{height haunch}	-				
A5	COMMAND reset(['collapse mechanism','thickness end plate','thickness web plate','height haunch','moment capacity'])	X	X	-	X	-		X	X	-	X	-				
A6	COMMAND repeat_table	X	X	-	X	-		X	X	-	X	-				
		R1	R2	R3	R4	R5		R6	R7	R8	R9	R10				

Fig. 6 Decision Table "Connection design"

In a situation where yielding of the end plate is the determining collapse mechanism there are two alternatives to increase the moment capacity of the connection. One can choose to include a thicker end plate or to include a (higher) haunch. Including a (thicker) web plate doesn't have an impact on the moment capacity in this situation.
In a situation where shear of the column web is the determining collapse mechanism there are also two alternatives to increase the moment capacity of the connection. One can choose to include a (thicker) web plate or to include a (higher) haunch. Including a thicker end plate doesn't have an impact on the moment capacity in this situation.



Moment capacity = 60kNm Moment capacity = 66kNm Moment capacity = 82kNm

Fig. 7 Functional equivalent connections

It can be concluded from the decision table that there are several alternative ways to reach the goal "good connection" [4]. Figure 7 displays three alternative connections, which are all good considering the goal "good connection", which actually means moment capacity greater than or equal to 60 kNm. Considering this goal these three geometrically different connections are functional equivalent.

Following a goal-oriented approach by analysing the decision process of a connection detailer trying to reach a specific goal, it is demonstrated that functional equivalent object-types are reconstructed. These functional equivalent object-types reveal that the attributes of these object-types can't be provided initially, because they are dynamic and conditional. They can only be acquired by reasoning about the goals of classification.

To demonstrate the relevance of the context in which a concept is reconstructed, figure 8 displays the concept "portal structure" in the context of a designer and in the context of a manufacturer.

A designer designs profiles and details connections between these profiles. A manufacturer manufactures assemblies, which generally consist of a profile and all the elements welded to the profile.

This example demonstrates that the reconstruction of concepts to define object-types, in this case the object-type "portal structure" depends on the context in which the concept is reconstructed. The relevant properties of a portal structure in a manufacturing context reasonably differ from the relevant properties in a design context. So, it is very important to define the context in which concepts are reconstructed.

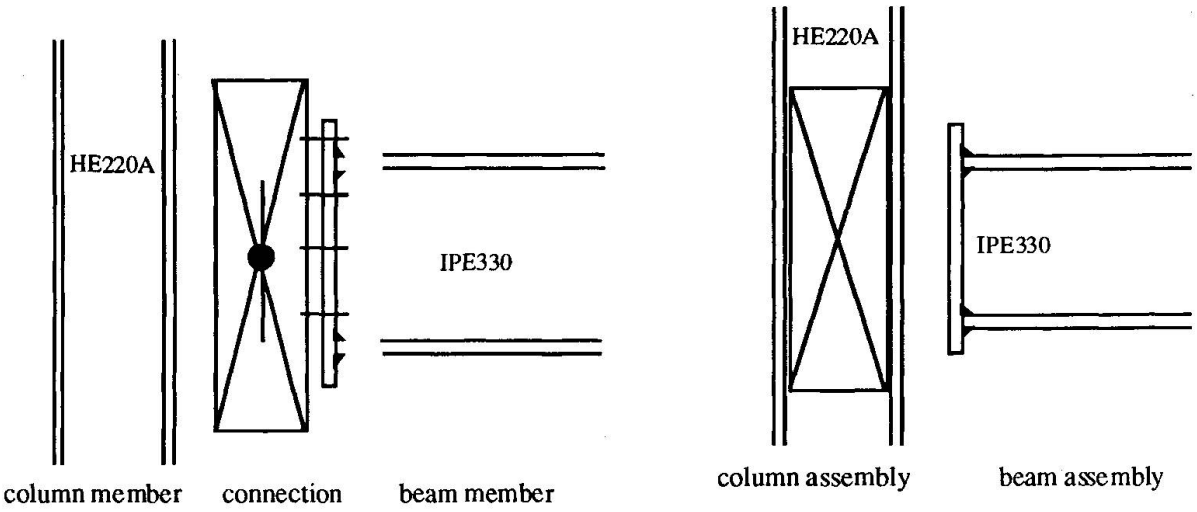


Fig. 8 Context dependent classifications of portal structure

3.3 Modelling steel structures using the theory of extensional classifications

In the theory of extensional classifications a conceptual model is developed by analysing the objects in reality which have to be modelled. Usually a data modelling technique like NIAM or IDEF1x is applied. In the IDEF1x-model developed within Eureka project EU130 "CIMSTEEL" [3] a connection is modelled as displayed in figure 9.

This model defines that a connection requires part-joints. And each part-joint connects two parts. In the first place this definition is rather generic. No difference is made between parts, like end plates, haunches and web plates. In the second place this definition is rather static. The model doesn't include information on the conditions in which a specific element is a part of the connection. The model doesn't include knowledge on functionally equivalent connections either. In reality a connection is a very heterogeneous object-type. The occurrence of elements is not static and can't be defined initially. Therefore it's very hard to describe connections by static and unconditional object-types and attributes. In the extensional approach this problem is bypassed by defining a generic model. The only general statement about connections is that they consist of an arbitrary number of joined parts. Consequently this is the only knowledge in the model.

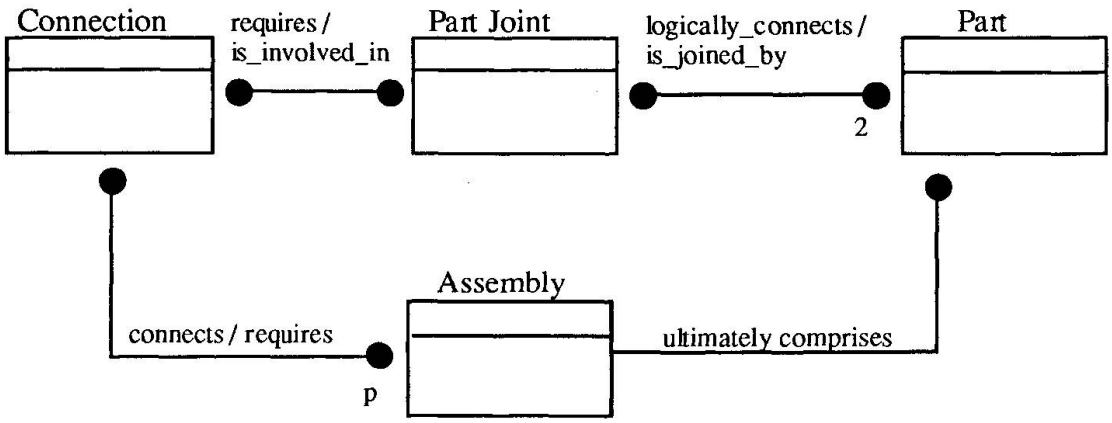


Fig.9 Fraction of CIMSTEEL product model

Concerning the design and manufacturing context, the CIMSTEEL model tries to combine these into one model. This is also shown in figure 9. A connection connects assemblies. An assembly comprises parts.
This combining of contexts shows that the existence of different contexts was recognized, but in order to be able to exchange information between different contexts one generic model was developed including aspects of both contexts. This makes the model rather ambiguous.

4. DISCUSSION

The paper introduced the theory of functional classifications as a basis for modelling steel structures. Usually the theory of extensional classifications is applied in this area. It was demonstrated by modelling a portal structure in steel that the theory of functional classifications should be preferred to the theory of extensional classifications, because:

- the goals of classification significantly influence the reconstruction of concepts
- only by reasoning about the goals of classification functionally equivalent object-types can be identified
- the context in which concepts are reconstructed has a significant influence on the classification process

By using the theory of functional classifications conceptual models are developed which contain knowledge directed towards goals of parties operating in the application area. The example demonstrated that the application area of structures is rather heterogeneous. Attributes are not static and unconditional, but dynamic and conditional. Dynamics and conditions can only be identified by following a goal oriented approach.

The theory of functional classifications used to develop conceptual models including knowledge of the application area, contributes to solutions in a number of research areas. These will be discussed now.

4.1 Conflicting goals

In the building and construction industry many parties are involved in the realisation of a building. Each party has its own goal and depending on that goal the party is only interested in a part of the building or in some aspects of a building. The occurrence of parties with different goals may easily lead to conflicts. It could happen, for instance, that environment requirements are in conflict with fire-safety requirements. Of course these conflicts can't be solved by an information system, but by following a goal-oriented approach in conceptual modelling it's possible to recognize these conflicts in an early stage. By developing goal-oriented models, both for environment and fire-safety, attributes occurring in both models with conflicting values can be recognized. By following a goal-oriented approach it could, for instance, be concluded that the material of a wall is an important attribute for the object-type 'wall' in both the environment model and the fire-safety model. If for fire-safety reasons a wall should be fabricated from material x and for environment reasons from material y, a conflict exists. The decision which material to choose can't be made of course by the system, but the system can report that a conflict exists, that has to be resolved.

This example also demonstrates the relevance of a context. With respect to fire-safety a wall of material x is classified as a good wall. With respect to environment a wall of material x is classified as unacceptable. This proves that the statement "good wall" is context-dependent.

4.2 Evaluation of representation formalisms

In chapter 3 a conceptual modelling approach was described, that can be used at the knowledge level. The knowledge level only consists of functional aspects of a system. Implementation aspects are not considered. Implementation aspects are considered at, what Newell calls, the symbol level [10]. At the symbol level object-types, reconstructed at the knowledge level are reduced to representation formalisms.

According to Newell:

Representation = Knowledge + Access

A representation formalism consists of structures to store knowledge and data and access mechanisms to operate on knowledge and data.
The conceptual model, developed at the knowledge level, can be used to evaluate representation formalisms [4]. Depending on the contents of the conceptual model, the most suitable representation formalism can be selected for implementation of the system.

Well known representation formalisms are record-based systems, and rule-based systems. A record-based system stores knowledge and data in records, which can be manipulated by a query language. A rule-based system stores knowledge and data in production rules, which can be manipulated by an inference mechanism, like forward chaining or backward chaining.

4.3 Integration of knowledge and databases

A great deal of research directed towards the integration of knowledge and databases, concentrates on the transfer of data structures and operations on data structures. In the past the statement could be heard: data is static and can be stored in records, knowledge is dynamic and should therefore be stored in production rules. Later it was realised that data is not always static, but sometimes dynamic and knowledge is not always dynamic, but sometimes static. One tried to solve this problem by using records to store production rules [6] or adding an inference mechanism to a database [16]. These are examples of symbol level integration of knowledge and databases.

The danger of these approaches is that the integration is realised without having a fundamental understanding of knowledge and data at the knowledge level. If only the functional structure of knowledge and data is considered, as worked out in chapter 3 for modelling of structures at the knowledge level, there is no clear distinction between knowledge and data. By following a goal-oriented approach using the theory of functional classifications, object-types consisting of attributes and constraints are reconstructed. These object-types consist of both knowledge and data and at that stage there is no need to bother about how to implement this, in a data-based system, a knowledge-based system or a combination of these. If in conceptual modelling too much attention is paid to representation formalisms, the functionality of the system can easily be adapted to the limitations of these formalisms. Consequently this leads to information systems, which are not really functional.

5. REFERENCES

1. BERG-CROSS, G. & PRICE, M.E., Acquiring and Managing Knowledge Using a Conceptual Structures Approach: Introduction and Framework; IEEE Transactions on Systems, Man and Cybernetics 19, pp. 513-527, 1989.

2. BRACHMAN, R.J. & LEVESQUE, H.J., What Makes a Knowledge Base Knowledgeable? A View of Databases from the Knowledge Level; Expert Database Systems, Kerschberg, L. (ed.), Benjamin/Cummings, Menlo Park, pp. 69-78, 1986.

3. CROWLEY, A.J., A "Walk-through" the LPM version 3.2, CIMSTEEL, EU130.2/LU/TP4, September 1992.

4. GELDER, J.T. DE & LUCARDIE G.L., Knowledge and Data Modelling in CAD/CAM-applications; Design and Decision Support Systems in Architecture and Urban Planning, to be published, 1993.

5. HENDRIKS, P.H.J., De Relationele Definitie van Begrippen: Een Relationeel Realistische Visie op het Operationaliseren en Representeren van Begrippen; Dissertation, Faculty of Policy Sciences, University of Nijmegen, 1986.

6. HERWIJNEN, J. VAN, HOUTEN, G. VAN, HOUTSMA, M. & ROMKEMA H., Implementatie van een regel-gebaseerd kennissysteem in een relationele database-omgeving; Informatie, Vol. 32, pp. 14-21, 1990.

7. HUIJSING, A.P., Een Expert System Shell voor Beslissingstabellen; TNO-rapport BI-92-156, October 1992.

8. LEVESQUE, H.J., Foundations of a Functional Approach to Knowledge Representation; Artificial Intelligence 23; pp. 155-212, 1984.

9. LUCARDIE, G.L., A Functional Approach to Realizing Decision Support Systems in Technical Regulation Management for Design and Construction; Design and Decision Support Systems in Architecture and Urban Planning, to be published, 1993.

10. NEWELL, A., The Knowledge Level; AI Magazine 2, pp. 1-20, 1981.

11. REITSMA, R.F., Functional Classification of Space: Aspects of Site Suitability Assesment in a Decision Support Environment; Dissertation, International Institute for Applied Systems Analysis, Laxenburg, Austria / Faculty of Policy Sciences, University of Nijmegen, 1990.

12. SMAGT, A.G.M. VAN DER, Definiëren en Relateren in Sociaal-Wetenschappelijk Onderzoek; Dissertation, Faculty of Policy Sciences, University of Nijmegen, 1985.

13. SMAGT, A.G.M. VAN DER & LUCARDIE, G.L., Decision Making under not Well Defined Conditions: From Data Processing to Logical Modelling; TESG 82, pp. 295-304, 1991.

14. SOWA, J.F., Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading etc, 1984.

15. STEELS, L., Kennissystemen; Addison-Wesley, 1992.

16. STONEBRAKER, M., Adding Semantic Knowledge to a Relational Database System, On Conceptual Modelling, Eds.: M. Brodie, J. Mylopoulos & J.W. Schmidt, Springer-Verlag, 1984.

17. STEENHUIS, C.M. & DOL, C., Geboute kopplaatverbindingen. Gebruik van NEN 6772 tijdens het ontwerpproces van een staalconstructie; Bouwen met Staal 103, pp. 3-7, 1991.

18. WAARD, M. DE, Computer Aided Conformance Checking; Dissertation, Delft University, 1992.

Design Reasoning with Cases and Intelligent Objects
Conception rationnelle avec des cas spécifiques et objets intelligents
Entwurfsprozess mittels Fallstudien und intelligenten Objekten

Gerhard SCHMITT
Professor
ETH
Zurich, Switzerland

Born 1953, M.Arch (1980) in Architecture, Univ. of California at Berkeley. PhD (1983) at the Technical Univ. of Munich. 1984 - 1988 computer-aided architectural design teaching and research at Carnegie Mellon Univ. Since 1988 Chair for Architecture and CAAD at ETH Zurich. Research interests: Intelligent Design Support Systems.

SUMMARY
This paper presents two new and promising developments in engineering and architecture design paradigms: designing with cases and with intelligent objects. Case-based design (CBD) derives new solutions by adapting existing designs. CBD avoids the problems of extensive search and combinatorial explosion. It helps to maintain the trade-offs and the quality of the case selected for adaptation. For case combination, which should be able to support innovative design tasks, intelligent objects are necessary. They are autonomous design agents of varying complexity which possess first principles knowledge, common sense knowledge, and interaction knowledge, among others. The paper describes experiences with CBD and expected contributions of intelligent objects.

RÉSUMÉ
Cette contribution présente deux développements nouveaux et prometteurs relatifs au paradigme de l'ingénierie et de l'architecture: concevoir à l'aide d'études de cas et d'objets intelligents. Case-based design (CBD) crée de nouvelles solutions à partir de solutions existantes tout en évitant recherche extensive et explosion combinatoire. CBD contribue à maintenir la pertinence et la qualité du cas retenu pour être adapté. Pour des combinaisons de cas capables d'alimenter des tâches innovatrices de conception, de complexité variable et possédant divers degrés de connaissance, entre autres, connaissance de base, de bon sens, des interactions. Cette contribution décrit des expériences avec CBD et de possibles apports des objets intelligents.

ZUSAMMENFASSUNG
Mit Fall-basiertem Entwerfen und dem Arbeiten mit intelligenten Objekten werden zwei Methoden vorgestellt, die den Entwurfsprozess von Architekten und Ingenieuren in der Zukunft verändern werden. Fall-basiertes Entwerfen konzentriert sich auf die Adaptierung guter bestehender Lösungen und vermeidet so viele Probleme traditioneller Computermethoden wie kombinatorischer Explosion und Erzeugung zu vieler Alternativen. Es unterstützt am besten Routine-Design Aufgaben. Das Arbeiten mit intelligenten Objekten erlaubt die Kombination von Fällen oder deren Teile und ist so für innovative Entwurfsaufgaben geeignet.

1. INTRODUCTION

Design reasoning with and based on *cases* has a long history in engineering and architecture. While experienced designers can rely on their own case base accumulated over the years, inexperienced designers rely more on generative methods and on external case bases [1]. Most architecture students learn to design using cases. Magazines present cases in varying depth, from images to detailed descriptions of the processes and the costs involved [11]. In graduate education, case studies are an effective teaching and learning tool. Cases help us to understand a building or a structure as a whole with all inherent trade-offs. They are the final and complex result of a successful design process. They do not necessarily reveal causal relations between design decisions and built results but they may lead to the discovery of such relations. Therefore, they are a very interesting topic of exploration in design computing.

Case-based Reasoning (CBR) and Case-based Design (CBD) are recently established research directions in Computer Science and Artificial Intelligence. They are the latest developments in a series of knowledge acquisition, knowledge representation and reasoning schemes which started with *prototypes*. The next logical step is the development of *intelligent objects* in a design environment which supports the previous representations. *Prototypes* are efficient for the solution of routine design problems, but are less useful for new or unexpected design problems because they have limited automatic adaptation capabilities. They require the ad-hoc definition of all relevant parameters. *Case-based representation* ideally stores all case-relevant information, including structured and unstructured data. It touches on all aspects of existing design and executes parameterization during the adaptation and combination process. Case input, case selection, and reasoning with real and complex design cases still pose formidable computational problems. CBD systems are already under development, *intelligent objects* will combine the advantages of the previous representations with the capability to automatically detect, for example, obstacles and possible violations of building codes and to react accordingly.

2. DESIGN REASONING

Design reasoning is the art of arriving at a design solution from an ill-defined problem specification. The final design solution contains trade-offs. Attempts in the past to see design merely as a decomposition problem or optimization process have failed, although support for this direction still exists [7]. The generation of design with grammars of shapes have been successfully implemented for the reconstruction of historic architecture but have not been widely accepted in practice.

One major reason for the failure of all previous attempts in design automation is a crucial property of human intelligence: the ability to maintain inconsistent solutions and partial solutions in parallel, judging involved trade-offs, while moving towards the design goal. No computer based method so far has offered a solution to this representational and reasoning problem. Researchers are able to analyze complete designs and give an explanation for a possible creation. However these activities have more to do with reverse engineering than with engineering.

Our recent research therefore focuses on a radically different direction. Using complete and existing design solutions and adapting them to a new problem appears promising if the following conditions are fulfilled:

- The new problem has a corresponding solution in an existing case. This requires a sizeable case base and appropriate access mechanisms.
- Appropriate abstraction and representation are available. Those include geometry, which is contained in the CAD model; functional and spatial relations, which can be expressed in a graph; structure, which can be expressed as a finite element representation in combination with geometric and graph representations; performance, which can be expressed as algorithms and constraints. All of these abstractions and representations symbolize important aspects of a completed design but are not necessarily compatible. Moving from one representation to another usually involves loss of information. Abstractions serve the purpose of allowing modifications at a high level without having to concentrate on details. This requires, in order to

guarantee an overall satisfactory solution, the final inspection of the result at the lowest possible level of abstraction, that is the geometric model. At the same time, working on the geometric model alone encourages most designers to focus too much on detail and may lead to a loss of overview and freedom for radical change.

- Appropriate adaptation mechanisms are available. This requires the presence of (i) dimensional adaptation which adapts the geometry of an object and of (ii) topological adaptation which is necessary when dimensional adaptation is unsuccessful.
- Appropriate evaluation methods are available. Every result of adaptation must be evaluated against the goal of the design process. The evaluation indicates whether a solution is successful or needs to undergo a new cycle of adaptation.
- Appropriate visual inspection techniques are available. Every representation of a design is an abstraction which is understood best by only one sector of the design team. The one common representation is the geometric appearance as the result of the design process. Therefore, every product of adaptation must be inspected visually to discover problems that can not be expressed in any of the other representations. For example, a circulation scheme which seems perfectly reasonable in diagrammatic form, might result in completely unsatisfactory architectural solutions.

This particular view represents *one* design reasoning paradigm. It needs extensive knowledge and infrastructure resources but promises complete and functioning solutions. Design reasoning using more traditional approaches such as generate-and-test will produce results that are less predictable and which may not be complete.

3. FROM PROTOTYPES TO INTELLIGENT OBJECTS

The scope of this paper is limited to the detailed discussion of case-based design and intelligent objects. In order to describe their capabilities and the differences to previously developed methods, brief definitions might be helpful:

3.1. Prototypes

Prototypes have been explored as a structure for representing design knowledge [4]. Prototypes are objects for which all parameters are known and pre-defined. Relations between parameters are expressed as rules or similar control structures within the object. Parameters and relations are subject to constraints. Frames are a feasible representation for prototypes.

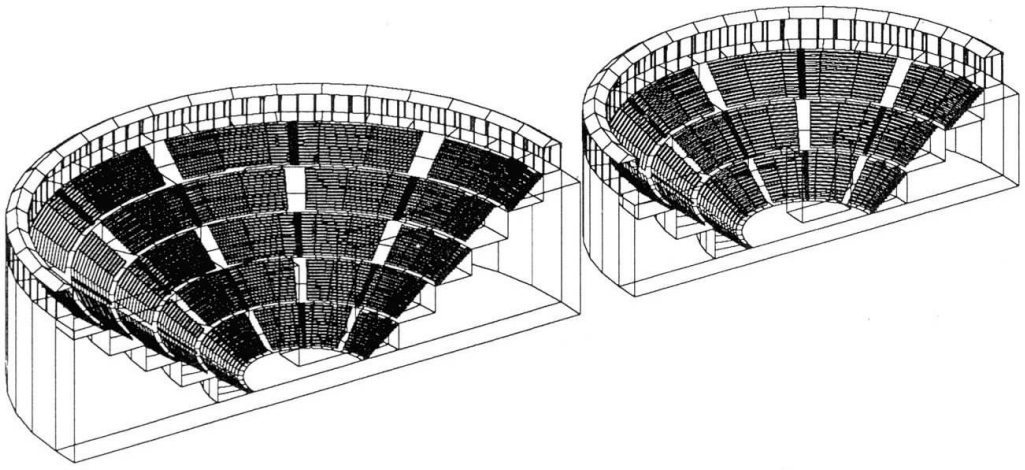


Fig. 1. Examples for objects created with prototypes. Automatic generation of a Roman theatre. Note the different number of seats which correspond to the different number of exits.

An advantage of prototypes is the rapid generation of complete alternatives that are guaranteed to satisfy expected performance criteria. A disadvantage is that the results are constrained and predictable. Therefore, prototypes are most useful for routine design rather than for innovative design.

3.2. Case-based Reasoning (CBR)

Case-based Reasoning is an effective knowledge acquisition and representation scheme for producing complete solutions with minimal search. It is being used increasingly in planning [5] as well as in industrial and military applications. Cases are stored as completely as possible in a case base. Cases are indexed according to their most relevant attributes. Most case-based reasoning applications focus on the activities of case selection and case indexing [8]. Case adaptation is a major concern for design applications.

An advantage of case-based reasoning is that cases need not be parameterized before use as is the case with parameterized objects or prototypes. The parameterization can occur during the reasoning process. Solutions are found quickly if a new problem has a near match in an existing case base or library. The disadvantages are similar to those of prototypes in that limitations exist for the generation of truly new solutions.

3.3. Case-based Design (CBD)

Case-based Design is the application of CBR techniques to design. As designs consist of complex objects, design cases can reach considerable size. Cases may contain, among other things, abstractions and representations of geometry, function, performance, circulation, and the structural system. Adaptation of complex cases is the major challenge in CBD.

Using prior cases is a common strategy for solving complex problems. CBD is different than the traditional rule-based approach for knowledge engineering in that it does not attempt to generalize and compile the knowledge before the problem is specified. Cases are regarded as storage of previous efforts and can be used as short-cuts for finding solutions. Applying this concept to computational processes, means using a given case as the initial state of the search process and searching through only the variations with great similarity to the initial case. The process is divided into case selection and case adaptation, each of which can be solved independently. If the initial case is properly selected, the search for a solution can be carried out much more efficiently. The separation of case selection and case adaptation also suggests a way of compromising that allows the introduction of manual guidance as the form of case selection to reduce the difficulty of machine computations. The implementation challenges of this approach are (i) the capability of recognizing potential solutions, if cases are to be selected automatically, and (ii) the capability to analyze the case and to derive case specific knowledge, for example what is good in this case and should be preserved during the adaptation or what is negative in this case and should be fixed. The final challenge is (iii) to modify the case so that the desirable features are not disturbed and at the same time variations that might solve the new problem may be generated efficiently.

The advantages of case-based design are similar to those of CBR. In addition, it is possible to integrate concerns related to the different representations and to apply adaptation mechanisms to the entire case, thus avoiding problems of combinatorial explosion or endless evaluation loops. Disadvantages are the difficulty of expressing design semantics and the problems of CBD in finding truly new solutions.

3.4. Intelligent Objects

Intelligent objects are autonomous agents that are able to show some kind of intelligent behavior. They can be structured and represented in a way similar to prototypes. In addition, they must "know" about the world surrounding them and thus contain some common sense knowledge. Intelligent objects can also contain first principles knowledge such as the laws of gravity. They are particularly suited for modeling in virtual reality environments where the constraints of two-dimensional screen interaction are absent.

The advantage of intelligent objects is their capability to perform meaningful configuration and simple design tasks without extensive human input. Because they contain common sense design knowledge,

they can help avoid most of the frequently repeated mistakes in design. Disadvantages are their completeness and the difficulties involved in modifying their behavior. They may appear as pre-packaged "black boxes" whose behavior must be accepted without the possibility of critique or change by the user.

3.5. Prototypes, Cases and Intelligent Objects: A Comparison

The use of prototypes and cases have a solid base in architectural education and practice. For computer implementation, the processes involved in reasoning with prototypes and cases must be known and formalized. If successful and fully understood, this will logically lead to the definition of intelligent objects which offer more design freedom while guaranteeing design quality.

Prototypes are most useful when all relevant parameters are known and the design problem is limited in scope, such as the reconstruction of historic architecture [9]. Roman theaters and temples can be reconstructed using a small set of parameters whose relations are known. This enables a programmer to build a tool that will correctly reconstruct an instance of Roman architecture if the appropriate parameters are input. Prototypes thus offer the fastest and "safest" results for well-defined routine design tasks.

In *CBD*, the selection of existing cases as a generating base for new solutions pre-defines the results as well. The reason is the necessary "closeness" of a new solution to an existing case without which the quality of the original case cannot be guaranteed. CBD is most appropriate for design in a sensitive context.

Intelligent objects do not require the extensive adaptation of entire design cases. Instead, they rely on the meaningful combination of known functional or physical design elements. Consistency and overall quality of the resulting design are only guaranteed by the knowledge inherent in the intelligent objects and by the guidance and combination process of the designer. Intelligent objects are a necessary prerequisite for successful case combination which is useful for innovative and creative design.

4. DESIGNING WITH CASES - CASE ADAPTATION

Since 1989 we have developed several case-based procedures for engineering and architectural design. After testing them with practical applications, we believe we have found a valid approach which can be applied in design and related fields. It consists of the following steps:

- Case and site description. This is the necessary first step in which the case and the environment are described to the machine. For this purpose, we developed the pre-processor Mod4 to model the case by inserting walls, doors, windows, rooms and contextual elements such as parcel lines, roads, buildings, lakes, and parks. Mod4 converts the input to AutoCAD format.

- Deriving knowledge from the case. The case input with Mod4 produces additional knowledge which is not visible in the graphical representation but which is present in the model. The system automatically identifies topological relationships among spaces by analyzing the model and its labels. Labels and positions of walls, doors, and windows are used to deduce required spatial adjacencies. Thus we build the different abstractions of a building into a case description which is used for later reasoning.

- Input of new design requirements. If a case must be adapted to programmatic changes as well as to a new context, the following steps are necessary: Specify the maximum and minimum sizes, areas and proportions of spaces (dimensional adaptation). If new spaces are to be added, specify the required adjacencies to other spaces. Indicate the search depth by defining how many rooms are to be reallocated (topological changes).

- Case insertion. The existing case is inserted into a new environment, most likely a new site. The insertion into a new site takes place according to a knowledge base that defines the positive or negative weights of direct links between spaces and contextual elements. For example, a living room facing south, a park, or a yard are positive weighting factors, whereas a bedroom facing a road or other buildings carries a negative weighting factor. The case is checked against its original site first. If a positive link is found, the weight is doubled, if a negative link is found, the negative weight is considered not so significant and thus is divided by two. The insertion is done by maximizing the sum of all links between the new site and the insertion of the case by rotating and mirroring the case.

- Dimensional adaptation. If the insertion satisfies the internal and external requirements, but not the dimensions of the new site, the first step is to dimensionally adapt the existing case. For this purpose, an evaluation detects discrepancies with the new site and converts these conflicts into parameters. The number of parameters can be large in the beginning and must be reduced to produce practical results. The process of dimensionality reduction [6] helps to reduce the parameters to a manageable number and also allows the integration of other design considerations, such as the structural system. If the dimensional adaptation is unsuccessful or produces visually unsatisfactory results, the topological adaptation process begins.

- Topological adaptation. We have tested different approaches for solving this problem. The first and most natural attempt was the use of case-specific rules. These rules fired whenever a conflict between the existing situation and the target situation was detected. As could be expected, the number of rules that were needed grew rapidly and degraded the overall performance of the system. A second attempt was the use of grammar of shapes and the application of neural networks to recognize changes in the grammar. The third and most promising approach employs the wall representation algorithms developed by Flemming. A derivation of Flemming's wall-representation is applied to search for topological variations [2], [3]. Linear programming is used to check the dimensional feasibility of each configuration. The case is used as the starting configuration. The process first removes the number of spaces that are to be reallocated, which already creates some alternatives. Based on these alternatives, spaces are inserted back in one after the other. If one configuration is found to be not acceptable, the entire search branch is discarded. If more than one positive solution is found, the user is prompted to select one to insert it into the new site.

  With this approach, we could solve some of the problems of specialized layout programs, such as the limitation to about ten configurable objects in a fixed layout. Rather than starting without a configuration, we take the existing case as a starting point for a re-configuration and can thus handle a larger number of objects. For example, the time required to solve a 30 space layout problem resulting from a topological change was about five minutes of CPU time on a Sparc Station 10. The algorithms produce a new layout which is then subject to evaluation and three-dimensional visual inspection.

- Evaluation and visual inspection. After the topological adaptation, the geometric results are displayed and evaluated. A visual evaluation of the geometric model is the last step. If the visual evaluation proves unsatisfactory, the above process is re-entered.

5. DESIGNING WITH INTELLIGENT OBJECTS - CASE COMBINATION

Case combination is necessary to achieve truly innovative and creative design solutions. The case combination system under study will be able to combine parts from different cases into a new, complete design. To achieve this goal, the individual parts must possess different types of knowledge in order to facilitate correct combination.

The knowledge is encapsulated in the design objects. In the beginning, there will be physical and functional design objects. *First principles knowledge*, such as the law of gravity, is most important for modeling physical objects. Physical building elements such as walls, beams, columns, and furniture will contain this knowledge. *Common sense knowledge*, which is lacking in most CAD

systems, will also be part of these objects. *Interaction knowledge*, necessary to define object and function interaction, will be encapsulated in physical and functional objects.

Design with intelligent objects must be real-time. This requires fast processors because resolving constraints and complying to rules are computing-intensive tasks. Relatively small intelligent objects, such as furniture and simple structural elements, will therefore be the first implementations. Once the intelligent objects become too complex, such as entire building sections including the corresponding technical installations, system performance will degrade rapidly and interactive use will be impossible. Predictability of the behavior will also be lost.

6. TWO NEW APPLICATIONS

We have applied the approach of case-based design to simple apartment layouts and to two residential buildings by the Swiss architects Campi and Pepsin [10]. Recently, we chose a large office building to test the mechanisms for integrating architectural and structural aspects and to resolve them concurrently. The two following examples will serve to demonstrate new aspects of the CBD approach: Emphasis on the context and learning from the actual process of case-based design.

6.1. Example 1: A house with eight rooms

Figure 2 shows the configuration of three building sites in a given context. The selected case is located on the west site and is to be used, after adaptation, for the two other sites as indicated by the arrows in the figure. However, the spatial needs of the new clients indicate that one more bed room has to be inserted for each adapted case. Therefore, dimensional adaptation will not solve the problem. Instead, after the input of new requirements, the program will generate topological alternatives within the constraints of the new sites.

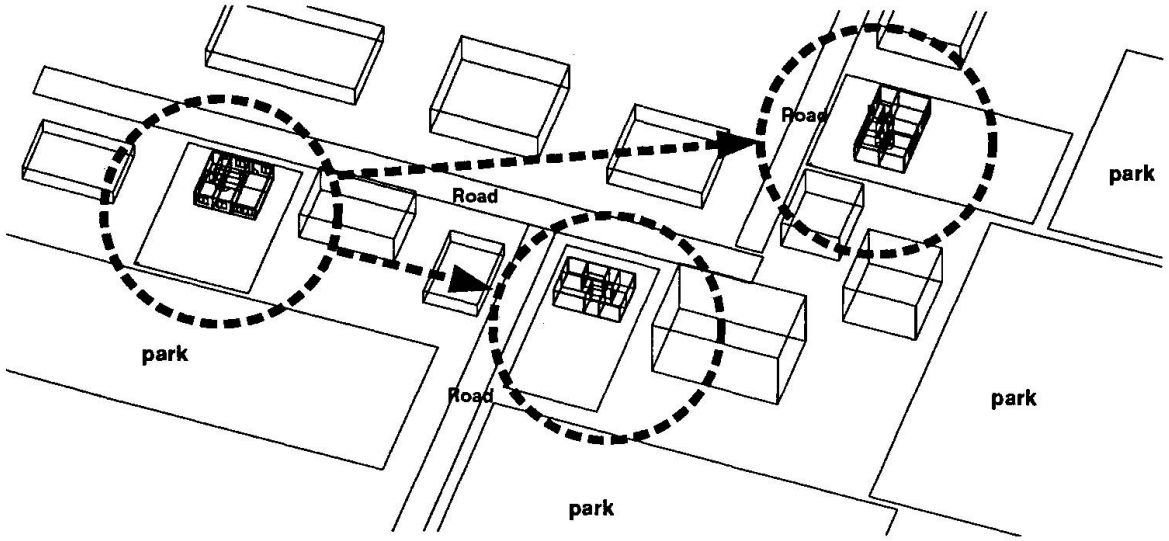


Fig. 2. The site with the original case and two case adaptations

Figure 3 shows the six alternatives for the second site (bottom center of the figure) generated by the program. The search process allows the utility room to be re-located. The reason is that this room has the least links to other spaces and to the exterior. One more bedroom is added. The alternatives are mirrored because there is a road on the west side of the new site: by mirroring the case, negative links between the bedrooms and the road are avoided. For the third site on the east, the same building plans are generated, but they are rotated by 90 degrees. The constraints introduced by the proportion of the site, the access to the road, and the position of the park are stronger than other relations that favor other orientations. In both cases, the user can select any one of the generated alternatives which appear on the screen. After the selection, they are automatically inserted into the site.

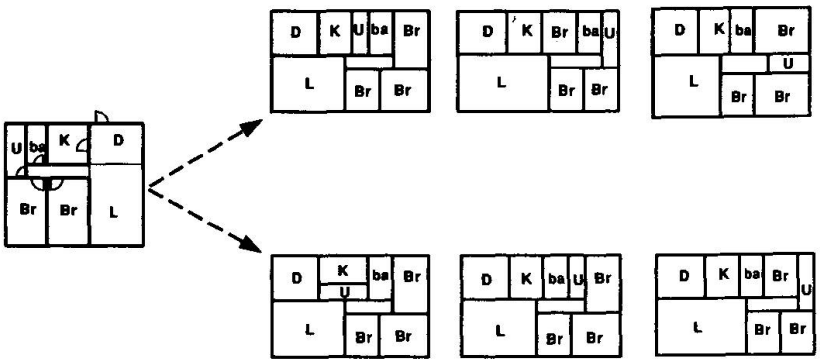


Fig 3. The six alternatives for the new layout with one added bedroom

6.2. Example 2: The timber frame barn

This experiment involved the design and construction of a small timber frame barn in order to detect strengths and shortcomings of the process we developed. The experiment provided a unique opportunity to close the feedback cycle. It consisted of the formulation of a theory, the computer implementation, and the application in practice. Moving from theory to implementation, a few simplifications had to be made, not all aspects of the project could be translated into a computer program. Testing the developed computer approach in practice showed the potential for using intelligent objects. It also revealed missing features in our CBD system.

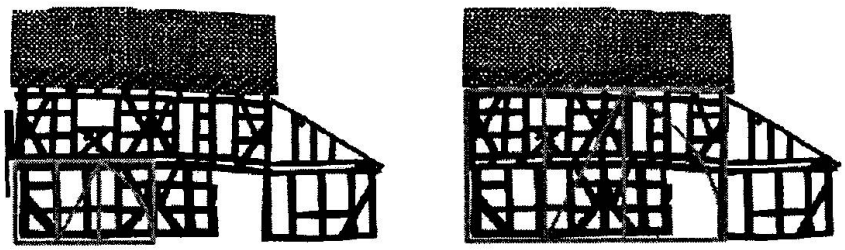


Fig. 4. The selected case and proportional constraints. The facades were topologically adapted to be used for the new case.

The task was to construct a timber frame barn. A number of goals, constraints, and wishes which became formalized only during the design and construction process, guided the design. Given were a program, a site with a 17th century timber frame residential building, and material. Based on these facts, the case-based design process started. The following list describes the procedure, the difficulties encountered in applying our existing CBD system (noted by CBD), and the potential of using intelligent objects in the process (noted by IO):

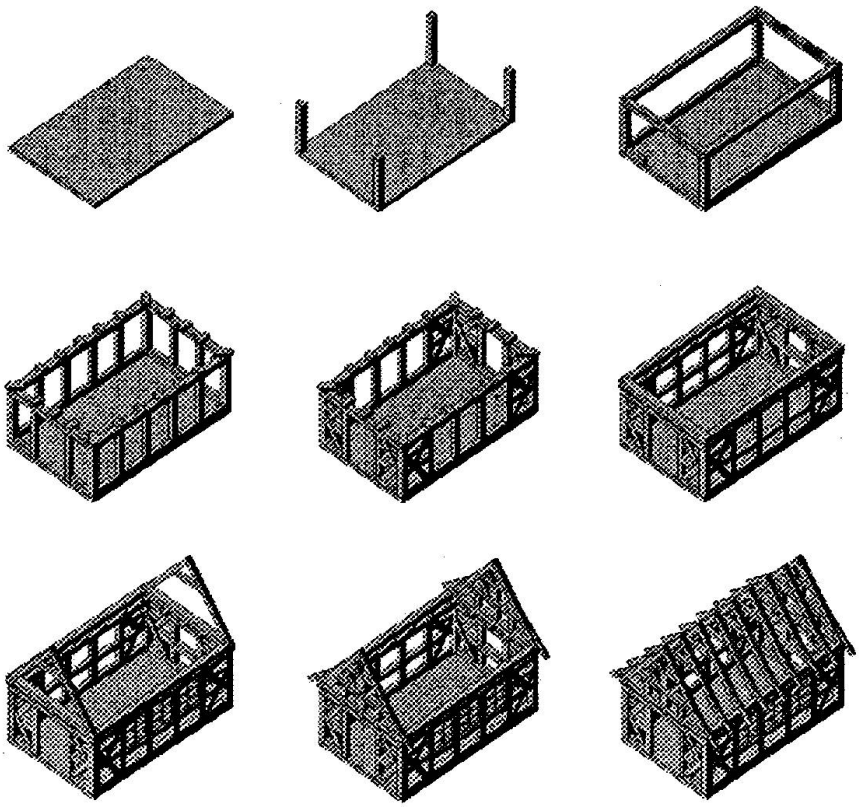


Fig. 5. Adaptation of the traditional construction process, applied to the already dimensionally and topologically adapted case.

- The program. Required was a storage space for a country house. The program resembles most likely that of a small rural barn.
  CBD: Our case base did not contain a building with this program.
  IO: The program will be an object with the appropriate attributes.
- The site. The new building should be close to the existing country house. At the same time, it should not obstruct the main facade of the existing building nor should it be placed too close to the existing structure. By specifying all known constraints, the location of the barn was defined.
  CBD: Parcel lines, existing building, forest, view, and street could be represented with Mod4. So far, we had not yet considered placing a new building on the same site as an existing case, nor had we dealt with non-rectangular sites.
  IO: The barn is an object which operates in an environment defined by geometry, attractive, prohibitive and avoidable features. The barn object has detectors which are sensitive to these features and change the allowed behavior of the object accordingly.
- Material. On site, 17th century oak beams were available with a maximum length of 5.6 m. This strongly suggested the use of the existing material, although some of it was in very bad condition.
  CBD: Our case base did not contain buildings in wood construction. There was no causal link between material and building type for the small scale of the problem at hand.
  IO: Intelligent objects are of little use in this respect, except attaching the request for a certain material as a constraint to the object.
- Construction. Having opted for the material and the site in close proximity of the existing building, traditional construction techniques would be the most reasonable to follow. After inspection of the construction details, it became clear that copying the traditional construction techniques would be too time consuming. Instead, an adaptation was suggested, replacing most of the wooden connections with steel profiles.

CBD: Our case base did not contain construction techniques. It is an important consideration, however, for the reconstruction of historic buildings or for new designs. Thus, a practical problem led to the expansion of the case description in the case base.
IO: Construction techniques and constructability could be made an active attribute of every building element. Existing wooden beams, for example, could request the appropriate fixings. They could disappear from the screen once the span they must cross becomes too large. They could warn the user if they are grossly oversized for the span they cover. They should still warn the user if too much load is added later. In the design of this barn, this would have been an important aid for the design of the roof construction.

- Proportions. All historic buildings in the area are constructed according to well established proportional rules. The common proportion in the elevations are the square and the golden section, the roof gables show angles from 1:1 to 1:2, the most satisfying being $1:\sqrt{2}$. All proportions can be explained by analyzing the design and construction process of the time.
  CBD: Proportional rules are part of our topological adaptation mechanisms.
  IO: design objects could vary within proportional constraints and react positively to desired proportions. For example, 1:1 and 1:2 could be the boundaries, 1:1.414 (square root 2) or 1:1.618 (golden section) could be used as defaults.
- Dimensions. The volume of the barn was as a consequence of the proportional and the material constraints.
  CBD: The existing case adaptation mechanisms are capable of achieving this dimensional adaptation. It reduces the original building basically to one space, which is a topological adaptation of a residence to a primitive hut. Topological adaptation in the other direction would be much more difficult.
  IO: Solving the dimensional constraints will be no problem for intelligent objects.
- Elevations. A direct dimensional adaptation of the existing facade was not possible. A topologically adapted facade passed the evaluation. Symmetry and mirroring were used to guide the topological adaptation process.
  CBD: Our case base did not contain elevations of this kind. However, once the dimensions were decided, the facade can be treated like a floor plan, thus making topological adaptation possible.
  IO: A facade as an intelligent object is a challenge because it has to interact with many other design elements: adjacent rooms, floor slabs, windows, materials, and structure.

The barn was constructed using the computer as the sole construction documentation tool. No paper or other traditional media were used, instead, two on site portables served as constant construction support. The main discovery was that CBD is useful for the building construction process as well.

7. CONCLUSIONS

Case-based design has proven its potential in adapting even complex buildings in theory. In practice, we could demonstrate with an experiment that CBD simulates a human reasoning technique that is found in almost any activity where experience and memory are important. CBD seems to provide a fundamental strategy for avoiding unnecessary re-generation of solutions. In the worst case, it may lead to uninspiring adaptations of existing design. Both the selected case and the adaptation process decide the quality of the final product. There still exist major problems in the representation and manipulation of complete cases which contain structured (e.g., CAD models) and unstructured (e.g., images, user responses, performance data) information.

Intelligent objects are essential elements for the realization of the idea of case combination. Scale and responsibility of the objects are critical parameters in implementation and interaction. Simple objects will be attractive for fast interaction, while complex intelligent objects will have to resolve too many constraints to be used in real-time. Our assumption is, that this limits the practical applicability of intelligent objects to well defined domains. Like cases, intelligent objects contain different representations, each of which is used for a specific design and interaction purpose. Our experiences with CBD have shown that it will be possible to work in real-time also with complex intelligent objects, if we operate on high levels of abstraction, such as diagrams.

8. ACKNOWLEDGEMENTS

The CBD project is funded by the Swiss Nationalfonds: NFP23, Artificial Intelligence and Robotics. I want to thank Boi Faltings, Ian Smith, Kefeng Hua, Simon Bailey, and Shen Guan Shih for the testing and implementation of the ideas presented.

9. REFERENCES

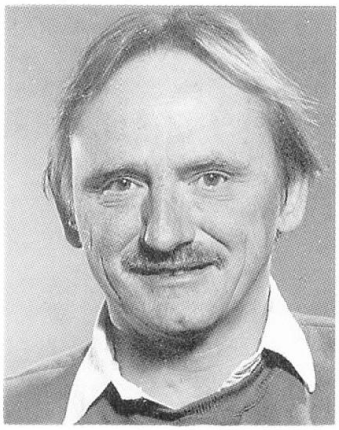
1. AKIN, ÖMER, *The Psychology of Architectural Design*, Pion, London, 1986.
2. COYNE, ROBERT, *ABLOOS: An Evolving Hierarchical Design Framework*, PhD thesis, EDRC 02-15-91, Carnegie Mellon University, Pittsburgh, 1991.
3. FLEMMING, ULRICH, R. F. Coyne, T. J. Glavin, and M. D. Rychener, *A Generative Expert System for the Design of Building Layouts*, Technical Report EDRC-48-08-88, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, 1988.
4. GERO, J., Maher, M.L. and Zhang, W.G. *Chunking Structural Design Knowledge as Prototypes*. The Architectural Computing Unit. Department of Architectural Science, University of Sydney, Australia. January 1988.
5. HAMMOND K., *Case-Based Planning*. Academic Press, Boston, 1989.
6. HUA, K., Smith, I., Faltings, B., Shih, S. and G. Schmitt. "Adaptation of Spatial Design Cases", in John Gero (ed.), *Artificial Intelligence in Design*, Kluwer Academic Publishers, The Netherlands, 1992, pp 559-575.
7. SIMON, HERBERT, *People and computers: their roles in creative design*, Keynote Lecture, Second International Conference on Artificial Intelligence in Design, Carnegie Mellon University, Pittsburgh, USA, June 22, 1992.
8. SCHANK, R. C., *Dynamic Memory: A Theory of Learning in Computers and People*, Cambridge University Press, Cambridge, 1982.
9. SCHMITT, GERHARD, "Rekonstruktion Aventicum", *AutoCAD Special*, Band 2, IWT Verlag, Germany, December 1990, pp. 124-133.
10. SHIH, SHEN-GUAN, "Case-Based Representation and Adaptation in Design", in G. Schmitt (ed.), *CAAD futures '91*, Vieweg, Wiesbaden, 1992, pp. 301-312.
11. "Théâtre de l'Enfance et de la Jeunesse, André Chavanne, Genève", *Werk, Bauen + Wohnen*, 12 Dezember 1992, p. 187.

Impact of User Demands on Knowledge-Based Systems

Impact des exigences des utilisateurs sur les banques de données

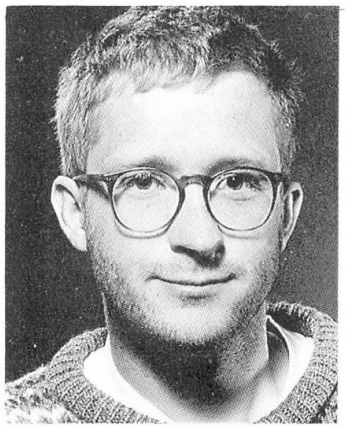
Einfluss der Benützeranforderungen an Entwurfsdatenbanken

Tom ANDERSEN
Assistant Professor
Techn.University
Lyngby, Denmark



Born 1954, M.Sc (1979) in Civil Eng., Technical Univ. of Denmark. Since 1989 member of staff, CAD Initiative. Research interests: knowledge-based systems, design theory, integration of software applications.

Niels Vejrup CARLSEN
Post doc
Techn.University
Lyngby, Denmark



Born 1963, M.Sc (1987) and PhD degree (1993) in Computer Science at the Technical Univ. of Denmark. Research interests: modelling of user interface software, user interface design, CAAD and virtual reality.

SUMMARY

When developing knowledge-based support systems for building design a query driven approach is generally considered very efficient. It provides a simple framework for knowledge acquisition and guides the modelling of the resulting knowledge-base. The identified queries define the structure of the user interface which relieves the designer from many considerations. However, this linkage gives certain system maintenance problems. Other issues in the design of knowledge-based design support systems arise from user demands such as flexibility of user interface and the existence of engineering strategies for logically separating the user interface software from the knowledge-base.

RÉSUMÉ

Le développement des banques de données basées sur les connaissances en matière de projet, implique une approche qui, partant des demandes des utilisateurs, est considérée efficace. Elle fournit un cadre à l'acquisition des connaissances et dirige la modélisation de la banque de données en résultant. Les demandes identifiées définissent la structure de l'interface et évitent un grand nombre de réflexions. Cette liaison soulève certains problèmes d'entretien. D'autres points de vue découlent des désirs de l'utilisateur, comme p.ex. la flexibilité d'accès à l'interface et l'assistance pour alimenter la banque de données par des connaissances personnelles. Ceci a pour conséquence de devoir débattre des stratégies de l'ingénierie du logiciel, afin de pouvoir séparer logiquement l'interface de l'utilisateur de la banque de données.

ZUSAMMENFASSUNG

Für die Entwicklung von wissensbasierten Datenbanken als Hilfsmittel zum Gebäudeentwurf wird der Ansatz, von Benützeranfragen auszugehen, als besonders wirkungsvoll angesehen. Es ergibt sich eine einfache Wissenseingabe und Strukturierung der Datenbank. Die Anfragen geben den Aufbau der Benützerschnittstelle vor und entheben den Ersteller vieler Ueberlegungen. Diese Verbindung führt zu einigen Unterhaltsproblemen. Weitere Gesichtspunkte ergeben sich aus dem Wunsch des Benützers nach Flexibilität im Zugriff und nach Unterstützung bei der Einspeisung eigenen Wissens. Entsprechend müssen Strategien des Software Engineering erörtert werden, damit die Benützerschnittstelle logisch von der Datenbank getrennt werden kann.

1 Introduction

BYGSYS is a knowledge-based system [9] which supports technical building design. The system supports the design of roof structures and it includes textbook knowledge and personal experience knowledge in its knowledge-base. The original aim of BYGSYS was to investigate to what degree experience knowledge could be utilized by an knowledge-based system [1].

Our work comprises a domain analysis and knowledge acquisition and elicitation based on a rather extensive survey of user demands [2]. Furthermore, we performed conceptual modelling, prototyping and testing/evaluation of the BYGSYS system derived from this analysis.

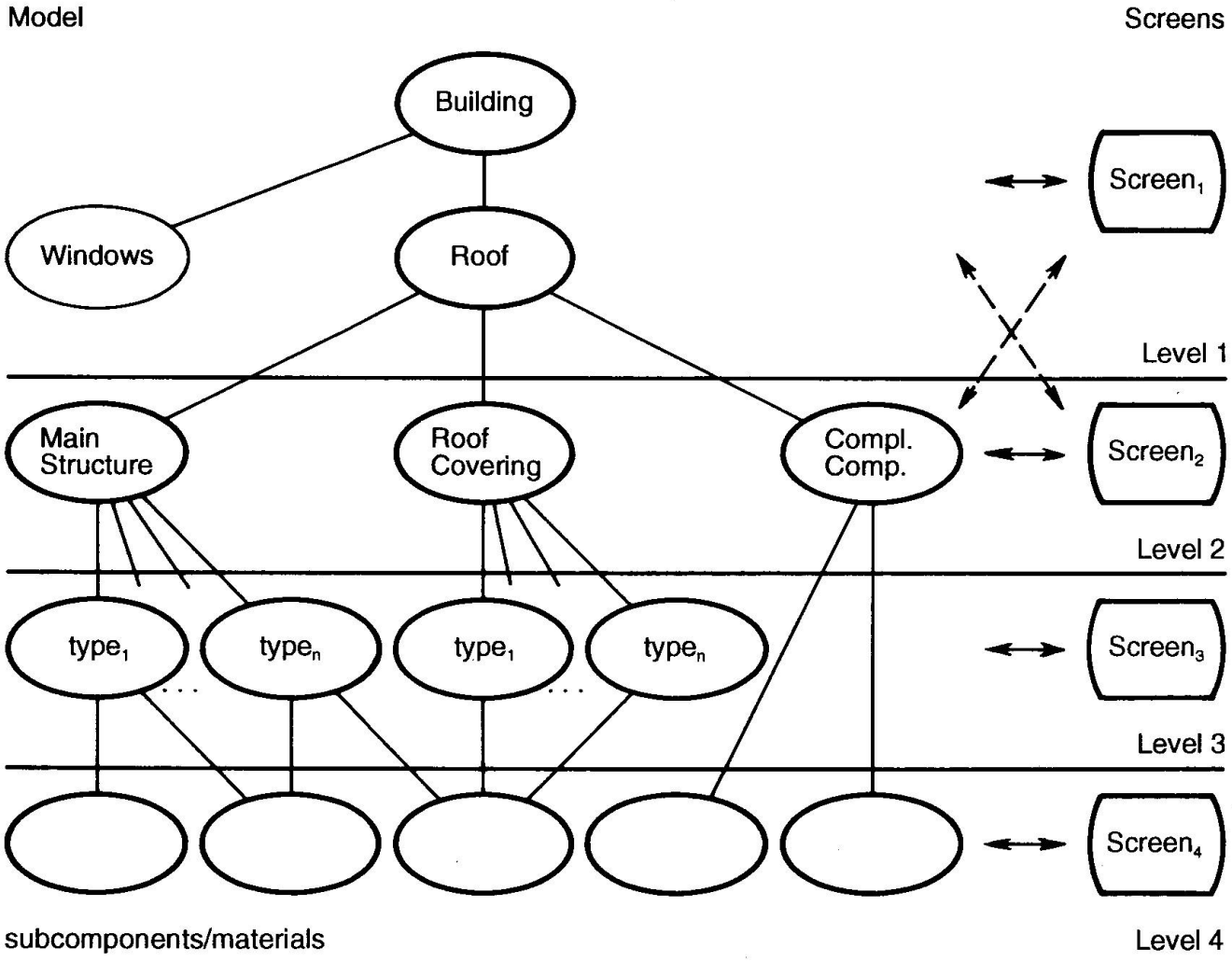


Figure 1. BYGSYS system structure and the corresponding screens.

In the following we will present in a little more detail the method with which we have designed BYGSYS. Thereafter, we evaluate the system functionality and its structure. This leads on to a discussion of software engineering issues that arise due to the development method and due to the impact of user demands.

1.1 Query Driven Knowledge Acquisition and Modelling

An analysis of the nature of technical design knowledge made it clear that the knowledge in this domain refers almost exclusively to physical objects, ie. the components of the building and its roof.

The following acquisition of knowledge where we initially focused on capturing *key queries*, ie. the most important issues for the potential users, emphasized the importance of identifying the components of the roof and their logical relations.

This query driven software design method applied to the BYGSYS system reflects the (simplified) way building design is normally performed. The process can be decomposed into a sequence of tasks, starting with the more abstract issues, ending with the detailed design. The sequence of queries was carefully investigated and approved by a number of professionals, who acted as users. The acquisition resulted in the model of the knowledge-base illustrated in figure 1. The links between queries and the related input/output screens (implementing the user interface) will be presented and discussed in the next section.

This BYGSYS model is equivalent to a product model [3] of the roof extended with some of the roof 'environment' such as the building. The model, shown in a simplified version, forms the core of the system and it can be regarded as a static view of the domain.

Having defined the static structure of the domain through an analysis of user demands - the query driven approach - we then gathered experience knowledge from professionals. This knowledge is represented in the dynamic part of the system and is implemented as procedures and rules on top of the static core structure.

1.2 User Demands

At this point it is important to recognize that user demands affect the system design in two separate ways. Our query driven design approach utilizes the fact that many of the user demands either explicitly or implicitly identify key queries that can direct the design of the knowledge base and the user interface to the system. These demands define *what the user wants to talk about* in the context of the given system.

Secondly, the user demands express *how the user wants to communicate* with this system. These demands of course have a strong influence on the design of the user interface to the system, but as we will see in the following they also have a profound impact on how the entire system should be engineered. The most important of these user demands are listed in table 1.

The system must not constrain the user. The sequence of decisions in a system session must reflect normal practice.
The dialogue should not be tedious and it must be flexible. The designer should be able to skip one or more inputs and possibly changé the default sequence of screens (if this makes sense in the context of the problem on hand).
The designer should be able to augment the system with personal experience.

Table 1. A summary of the most important user demands

As discussed in the above, we found that the sequence of user queries in this domain is (almost) fixed and that it reflects the order of decisions taken in a building design process. We therefore designed input/output screens according to the demands such that each of these correspond to one key query. Thus, the levels in the model and the sequence of screens are strongly connected as the system structure in figure 1 shows.

Please notice that screen 1 and 2 can be chosen randomly, that is, the users can change this particular part of the sequence. This partly satisfies the second user demand of table 1.

The design of the individual screens also supports many of the user demands. The system was designed to avoid too many screens and thus too many questions. A feature is the support for blank entries, so the user only has to concentrate on those aspects relevant to the actual case at the given time of the session. Figure 2 shows one of the major screens illustrating this.

1.3 Analysis

Our knowledge acquisition and modelling strategy has proven very successful. The query driven approach is effective, it speeds up prototyping while retaining the quality of the final system. Furthermore, the queries delimit the domain knowledge and helps focus attention. Using this approach helps avoid the gathering and subsequent implementation of large amounts of irrelevant knowledge. However, we must emphasize that identifying key queries is a time consuming process which has to be performed very carefully.

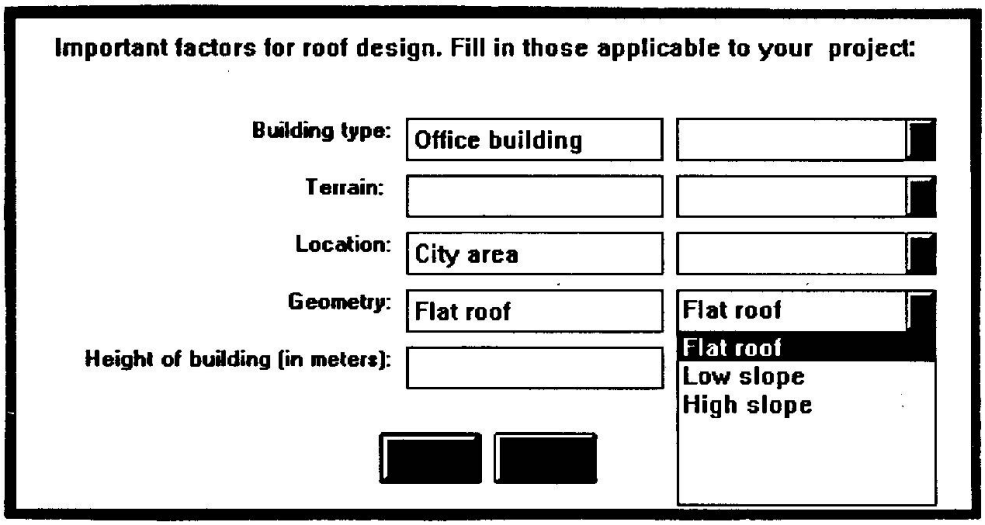


Figure 2. Screen design

The separation between the static model and the associated dynamic experience knowledge supports augmentations of the knowledge-base. More rules can easily be added to the system without revising the entire model. So, user defined knowledge (see table 1) is accommodated.

BYGSYS lets the user augment the system with his/her own rules in a rather simplified way. The rules can be added to the system without any programming skills, but the rules are 'passive' in that they do not interfere or interrupt the actual knowledge processing of the system. They can only support the user during a session, by sending messages and/or warnings to the screen concerning decisions to be made at the actual time. This feature was judged by professionals to be very useful.

In general the BYGSYS user interface was rated as satisfactory by a chosen group of professional users (ie. it met most of their demands). This success naturally lead to investigations into how the knowledge domain could be expanded so that the system could address larger parts of the technical building design process and how some of the more advanced user demands (in particular the possibility for further user defined augmentations of the knowledge base) could be better accommodated.

2 Software Engineering Issues

Some of the more general user demands dealing with flexibility of user interface and with the provision for user defined system augmentations enhance a basic software engineering problem: how to design a flexible and maintainable system. BYGSYS attempts to accommodate these demands to some degree - but not fully. In the following we analyse what impact these demands have (or should have) on the development of design support systems.

The software structure of the existing BYGSYS system does for example not facilitate the introduction of new key queries as discovered by a new analysis of user demands or developments within the field of expertise covered by system. Also, user demands that radically change the sequence of screens can cause system maintenance problems.

2.1 Maintenance of Knowledge-Base

When designing software, the demands of the users are very important, in particular when dealing with interactive decision support systems. The BYGSYS system is heavily influenced by user demands. This is only natural since query analysis was a major factor in the development process. But also user's expectations on screen design and other user interface aspects had an important impact on the system structure.

We have been able to satisfy these demands within BYGSYS. But this is under the condition that the hierarchy of major elements (classes like 'roofing material') is correct and exhaustive and thus that the levels of 'communication' shown in figure 2 are the only ones to address in the domain. That is, in a somewhat narrow domain like roof design the functional core can be defined correctly, once and for all, if this is done carefully.

However, if the knowledge of a domain is more general and/or comprehensive and widespread, a pure query driven approach can be dangerous. It will be very difficult to ensure that all queries are in fact captured before an initial design and system implementation has been completed. It is very likely that unexpected demands will be discovered after this implementation and this might require quite substantial system modifications.

If a new key query is introduced, more functionality of the system is required. The user needs access to a level of knowledge which is not in the current basic structure of the system. In the BYGSYS case we might imagine a future need to include knowledge about a brand new class of roofs with a different breakdown as compared to traditional roof structures. This would cause an extra layer in the model, as visualized in figure 3.

BYGSYS propagate knowledge down the hierarchy in a fashion similar to knowledge propagation in the building design process. Overall decisions are preserved through the stages of design and are strongly connected to detailed decisions. A choice of type of roofing such as tile will for example affect later decisions on many detailed aspects of the roof. BYGSYS reflects this process, and general decisions will have an influence on lower level decisions later in the design session.

One can think of this as knowledge links. High level knowledge, represented in 'high level' objects, are linked to lower level objects. The choice of tile roofing will for example propagate knowledge and information to the "material" objects of the model. These knowledge links are among other things essential in a system that attempts to eliminate trivial and redundant user input by introducing relevant and intelligent options for the users decisions.

BYGSYS has several of these knowledge links which presents a problem when modifying the system by introducing extra layers. All the links have to be maintained when cutting and pasting yet another key layer into the model. Depending on the actual case, it may require a substantial amount of work to introduce this extra key query.

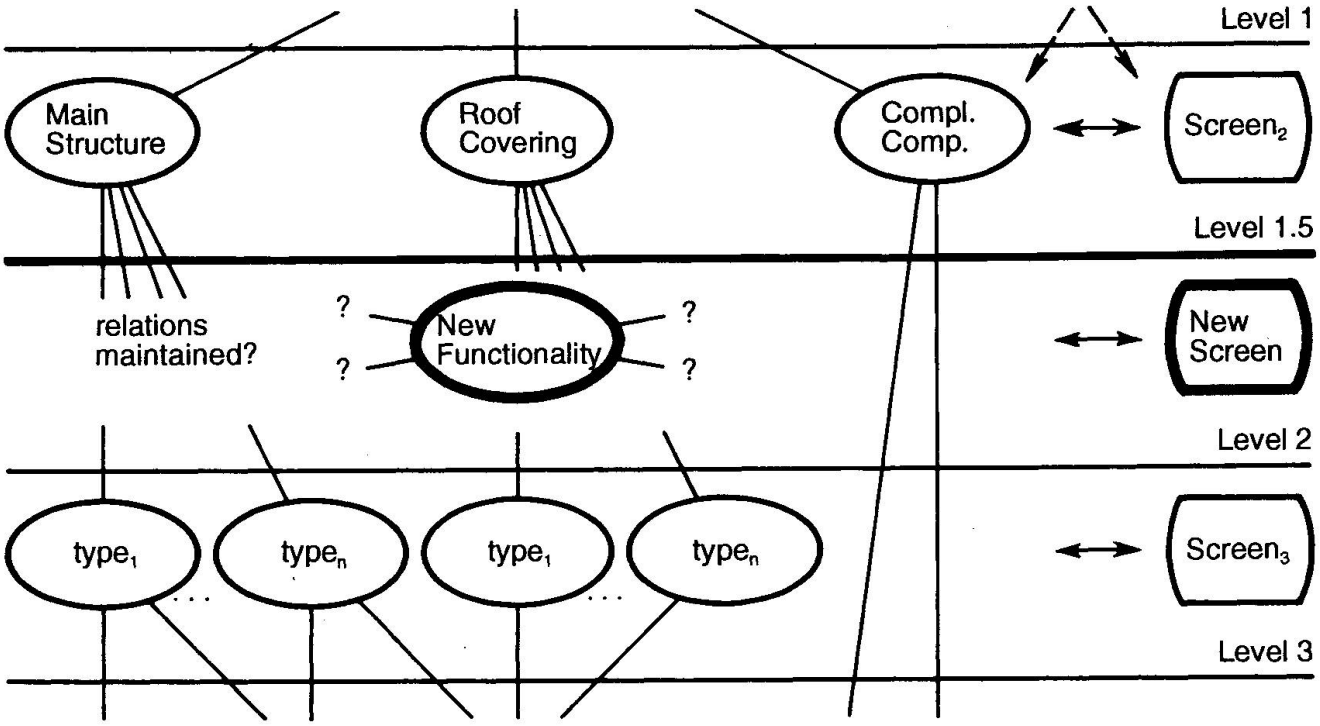


Figure 3. Adding key queries and screens to the system.

2.2 Flexibility of User Interface

If users demand that the dialogue flow, the sequence of screens, be radically changed or be made more flexible, these links again present a problem since all knowledge is propagated top down in the system. If for example we wish to comply with the fact that users may wish to work bottom up, or 'middle out' in certain design sessions we need to change the dynamics of the knowledge base and the screens must be able to deal with the fact that they may be invoked under different circumstances.

So, both the underlying knowledge-base and the associated user interface must be revised in response to these types of user demands. This is complicated by the fact that the user interface is tightly coupled to the structure of the knowledge-base. As illustrated in figure 4, some of the objects in the knowledge-base (UI objects) contain both functionality defined on the given type of knowledge and code dealing with the user interface. Also, calls to the user interface is often a product of the knowledge itself.

2.3 User Driven Augmentations

Both of the above problems have been discussed under the assumption that any changes to the system are the responsibility of a system manager/programmer. However, one general user demand was that user's should be given facilities for defining such changes themselves. It should be possible for users to change the dialogue flow of the user interface and they should be able to augment/modify the underlying knowledge-base. This makes things worse. If the system is not easily maintained by a system programmer then imagine trying to construct an easy to use tool for letting users do the job themselves!

3 Separation of User Interface Software

BYGSYS' problematic lack of separation between its user interface code and its functional core, ie. the static knowledge-base plus rule dynamics, is actually evident in most knowledge-based design support systems. The reason being that software development traditionally focuses on the design of system functionality. The input/output actions defining the system's user interface are seen as (unfortunate) side-effects of computation.

This tight coupling has important advantages such as efficiency of system development and of run-time performance. However, as shown, it does give certain system maintenance problems. These problems are further enhanced by the fact that the development of a user interface for a given system must be an iterative prototyping process due to the diversity of human users and of their work situations.

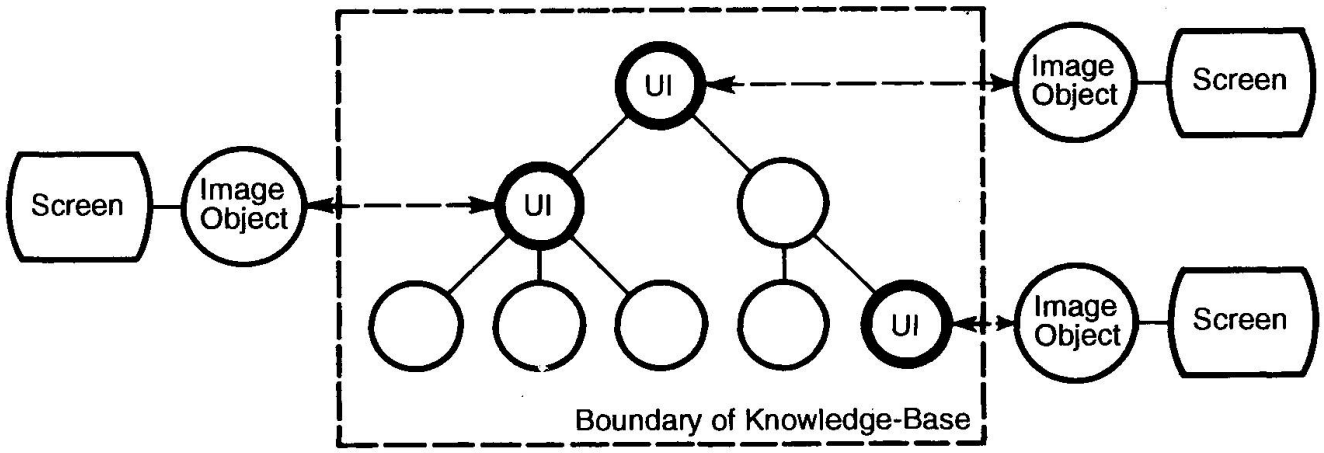


Figure 4. The coupling between knowledge-base and user interface. Within the shell used [7], 'image objects' handle the linkage between screens and knowledge-base objects.

First of all, even minor changes to the flow of dialogue, like the ordering of screens in our BYGSYS interface, will most likely require substantial changes to the entire system. This makes prototyping prohibitively expensive and obstructs maintenance and augmentation in the later stages of the system life cycle. Secondly, the fact that there is no separate explicit and manipulable representation of the user interface (except for the layout and local interaction control handled within the screen controlling image objects) makes it very difficult to construct tools for user defined adaptation and augmentation of the system.

3.1 The Separation Strategy

This motivates the goal of placing the user interface software of an interactive system into a separate module. This module must be used by the functional core for any desired communication with the user. The most important design goal here is to achieve dialogue independence for the functional core of a system so that user interface design modifications do not affect it as long as they do not require additional computations. Also, the user interface software should be isolated from changes to the functional core that do not affect the end users view of the system.

In addition the communication bandwidth required between the two modules should be minimized as much as possible. This will further reduce the interdependencies that compromise maintainability and it will allow the system to be integrated more easily within a networked environment.

The user interface software & technology (UIST) community has been researching this issue intensively for the last 10 years [6]. The main problem with achieving separation is the need for *semantic feedback* [4]. Feedback on the semantic effects of user inputs requires the user interface to have knowledge of or access to the state of the functional core. In more advanced graphical user interfaces where the user can manipulate data directly and where feedback on this must be given continuously, this becomes a serious problem.

In the more traditional query based interfaces that are most applicable to the design of a design support system like BYGSYS this is fortunately not a real problem. Here it is sufficient to exchange semantic information between the user interface and the functional core at closure points in the dialogue. This means that it should be possible to come up with a sensible development strategy for such systems!

The strategy for separating user interface from functional core must deal with 1) how the inevitable exchanges of information across the separation interface are to be coordinated and 2) how information is distributed between the two modules so as to minimize this communication. These two aspects are covered by the *global control model* and the *information distribution model.*

3.2 The Global Control Model

The *global* control mechanism governs the flow of control between the two domains whereas *local* control mechanisms handle dialogue flow within the user interface domain and the algorithmic flow within and across functions of the functional core. The global control model must define which component is in charge of the information exchange and whether this control may shift between components. Three distinct strategies have been proposed [4]: the *computation dominant control*, the *dialogue dominant control* and the *mixed control* models.

Systems like BYGSYS that employ the computation dominant control model let the user interface software be controlled by the functional core. The user interface is implemented by a set of abstract interaction devices controlled by the functional core. The *image objects* used in the BYGSYS implementation for controlling the screens exemplify such abstract interaction devices.

The computation dominant model has two basic problems. First, the *overall dialogue sequencing* is only implicitly defined by the local control flow of the functional core. This compromises dialogue independence and consistency and it furthermore discourages modifications of a given user interface design to a functional core.

Second, user interfaces constructed with systems employing this separation strategy tends to exhibit a *highly moded structure*. The functional core decides when the user may input a certain piece of information according to its internal control flow, eg. when the problem solving mechanism needs new data to continue processing the knowledge base. Since its execution is temporarily stopped until this information is returned a moded structure is forced onto the dialogue.

The dialogue dominant control model is a better approach to system design. Here the functional core is accessed by a set of functions controlled strictly by the user interface module. This avoids the above problems but introduces new ones. These functions must be atomic. During their execution they may not communicate with the user, eg. to retrieve missing data or to communicate error conditions arising within the functional core.

The mixed control model is therefore (in principle) the best solution. This model extends the dialogue dominant model by allowing functions in the functional core to call the user interface during execution. This is the model assumed by most contemporary user interface toolkits such as the X toolkit [10].

3.3 The Information Distribution Models

The second and generally most tricky problem in defining the separation interface is to specify the distribution of data across the interface so as to minimize communication bandwidth while still achieving dialogue independence.

This *information distribution model* must specify the kinds of information residing in each module, how and at what level of abstraction the two domains exchange information. It must define how output objects from the functional core are handled within the user interface and must decide whether or not user interface mechanisms that handle semantic feedback have direct access to the internal data of the functional core.

As briefly discussed in the above, this model is not critical in the design of a query/answer system like BYGSYS where information only needs to be exchanged in relatively small units (slot values in knowledge objects) and at clearly identified closure points (each time a level in the hierarchical knowledge structure is processed). Nevertheless, the approach taken will have an effect on the maintainability of the system.

BYGSYS employs the *shared data* information distribution model. All information within the functional core that is relevant to the user interface is placed in a shared data structure. In casu BYGSYS this is actually the knowledge base itself, the image objects can retrieve slot values in the knowledge base directly (eg. for displaying legal choices in a screen). Also, they have direct access to the rules and procedures triggered by the various choices made by the user within the screen.

This approach of course facilitates the user interface management of semantic feedback and it relieves the functional core from considering user interface aspects such as text formatting etc. However, the model obstructs system maintenance - all the direct references must be maintained.

The *distributed data* information distribution model seems to be a better approach. It tries to address these problems by suggesting that the internal data be distributed. The functional core defines a relevant view of its internal database through a set of mapping functions. Updates in the view are explicitly communicated between user interface and functional core. Low bandwidth across the separation interface can be achieved so this is an ideal model.

4 Guidelines & Conclusions

This discussion of separation strategy leads to some guidelines for constructing maintainable, flexible and user augmentable design support systems. Traditionally, a knowledge acquisition phase is followed by a conceptual modelling phase whereupon the system can be constructed.

4.1 Guideline

The query driven approach provides a good starting point for performing domain analysis and knowledge acquisition. It may also be put to good use in the context of the actual system design. However, it is at this point that we should take the above separation issues into account.

Instead of focusing solely on the construction of the knowledge-base and postponing most user interface considerations, these should be developed more or less in parallel. This is actually supported by many contemporary knowledge-base shells where eg. image objects and screens can be defined and used within the scope of the shell.

However, the shells do not directly support the construction of systems with a clean separation between user interface and functional core. The overall flow of dialogue and the presentation of data must still be handled within the functional core.

We recommend the introduction of a separate *linkage module* [5] between the knowledge-base and the user interface functionality. This module must contain an explicit and manipulable representation of the dialogue, it must implement the chosen global control model and information distribution model.

This linkage module may just be a conceptual entity - representing the fact that the designer does strive for separation - or it may be realized as a physical entity. Intelligent user interfaces [8] may employ user modelling techniques in order to adapt to users. Others may embody handwriting/voice recognition and/or natural language understanding. In these contexts a *blackboard architecture* with three separate knowledge-bases could be a sensible implementation strategy.

4.2 Conclusions

One important conclusion of the above is that not only does the user demands impact knowledge-base design and user interface design but they also impact the approach to software design that must be taken when designing the given design support system.

User (query) driven modelling and design does without doubt imply incremental prototyping. However, in the case of larger systems and/or complex knowledge domains, this is only feasible if care is taken to have optimal separation between the user interface and the functional core. An initial prototype can easily be developed without separation (as supported by most shells) but later revisions of this design will most likely be prohibitively complex.

So, user demands must be taken seriously when designing decision support systems. We recommend user (query) driven knowledge acquisition and modelling but combined with an overall strategy for software separation.

5 References

1. T. Andersen; A. Gaarslev, Building faults - preventing future faults by utilizing past experience, in: V.Ireland (ed), Proceedings of CIB90 vol.7, 1990.

2. T. Andersen, BYGSYS technical report, Dept. of Construction Management, June 1992.

3. B.-C. Björk, Basic structure of a proposed building product model, CAD-journal, vol 21(2), March 1989.

4. N.V. Carlsen, Towards a Common Context for User Interface Management System Design, in: C.Unger & J.A.Larson (eds) Engineering for Human-Computer Interaction, to be published by Elsevier Science Publishers, 1993.

5. G. Cockton, A New Model for Separable Interactive Systems, Proceedings of INTERACT'87, Elsevier Science Publishers, 1987.

6. H.R. Hartson; D. Hix, Human-Computer Interface Development: Concepts and Systems for its Management, ACM Computing Surveys, volume 21(1), March 1989.

7. Intellicorp, User's Manual, KAPPA-PC, 1991.

8. J.W. Sullivan; S.W. Tyler (eds), Intelligent User Interfaces, ACM Press, Addison-Wesley, 1991.

9. D.A. Waterman, A Guide to Expert Systems, Addison-Wesley, 1986.

10. D.A. Young, The X Window System - Programming and Applications with Xt (OSF/MOTIF Edition), Prentice-Hall, 1990.

A Memory Organization for Heterogeneous Design Knowledge

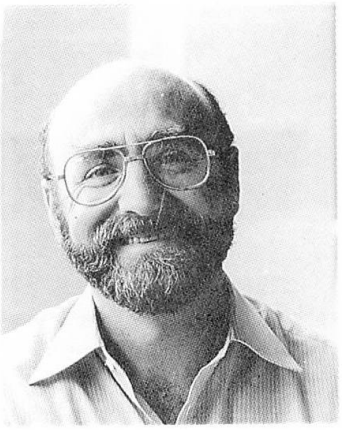
Organisation des données pour des connaissances hétérogènes en matière de projet

Datenverwaltung für heterogenes Entwurfswissen

Weiyuan WANG
Doctoral Student
University of Sydney
Sydney, Australia

John GERO
Prof. of Design Science
University of Sydney
Sydney, Australia

Weiyuan Wang received his BS degree in Computer Eng. from the Chongqing Univ., China, in 1983. Currently a doctoral student, his research interests include machine learning in design, knowledge acquisition and representation.



John Gero holds degrees in engineering, building science and architecture. He has been a visiting professor at Universities in the USA, UK and France. His current research is in artifical intelligence in design.

SUMMARY

Design prototypes and design cases are two disparate approaches to the representation of design knowledge. Design prototypes are a schema for representing compiled knowledge while design cases are a schema for representing situated knowledge. This research attempts to integrate both knowledge in a single system. A generalization-based memory organization for heterogeneous design knowledge is presented in this paper. Machine learning techniques are used to generalize knowledge-rich design schemas which function as both a knowledge representation schema for storing generic design cases and a memory organization facility for indexing design cases and design prototypes.

RÉSUMÉ

Les prototypes de projets et les études de cas spécifiques sont deux approches incompatibles pour représenter les connaissances en matière de projets: la première transmet un savoir par assemblage, tandis que la seconde procure un savoir en fonction de la situation. L'objectif de la recherche consiste à intégrer les deux domaines de connaissances en un système unique. Les auteurs présentent une organisation de données basée sur la généralisation pour les connaissances hétérogènes en matière de projets. Des moyens didactiques mécaniques servent à condenser des schémas de projets riches en informations, qui satisfont aussi bien à une fonction de stockage des cas d'études considérés comme spécifiques, qu'à une fonction d'organisation de données.

ZUSAMMENFASSUNG

Typenentwürfe und Entwurfsstudien sind zwei inkompatible Arten, Entwurfswissen darzustellen: Erstere vermitteln zusammengetragenes Wissen, während letztere situationsbedingte Erkenntnisse wiedergeben. Die Forschung versucht beide Teilbereiche in ein System zu integrieren. Als Beitrag dazu wird eine auf Verallgemeinerung bauende Datenorganisation für heterogenes Entwurfswissen vorgestellt. Unter Nutzung maschineller Lernfähigkeit werden informationsreiche Entwurfsschemata kondensiert, die sowohl eine Funktion der Ablage erdachter Entwurfsfälle als auch der Datenorganisation erfüllen, indem auf Entwurfsstudien und Typenentwürfe verwiesen wird.

1. INTRODUCTION

Design is complex and a designer usually utilizes several types of knowledge in solving a design problem. The design knowledge can be general or specific, formal logic or heuristic rules, generalized from experience or situated in design cases, etc. To integrate different types of knowledge to create more powerful systems is an important issue in the current research on knowledge-based systems. Although the integration of different types of knowledge or knowledge representations has the same importance as the integration of reasoning paradigms in the development of such systems, most research has focused on the integration of different reasoning paradigms while there has been little on the organization of heterogeneous design knowledge. This paper presents one approach to the memory organization for heterogeneous design knowledge and addresses the relevant issues.

Two types of widely-discussed design knowledge are design prototypes and design cases. Design prototypes are a schema for representing compiled design knowledge while design cases are a schema for representing situated design knowledge. Design prototypes are efficient in generating and evaluating design alternatives because of their richness of knowledge while design cases are powerful in providing a good set of initial values for the instantiation of a design prototype [12]. Case-based design needs to be supported by domain knowledge such as qualitative models [3], decomposition models [8] and design prototypes [11]. On the other hand, prototype-based design can be enhanced by the design knowledge situated in design cases [10]. This research attempts to integrate both kinds of knowledge in a single system in order to complement the strengths of each. To connect heterogeneous knowledge and link multiple representations, we must have an efficient way to organize the memory. Since the knowledge is acquired with different methods and from different sources, we must also have a mechanism to maintain the consistency of memory. We have proposed an integrated learning model [13], which has been further developed into a generalization-based memory organization for heterogeneous design knowledge. A knowledge-rich design schema has been introduced which functions as both a knowledge representation schema for storing generic design cases and learned knowledge, and a memory organization facility for indexing design cases and linking them to related design prototypes. Machine learning techniques have been applied to generalize the schemas and organize them in a hierarchy. The knowledge in the schemas is in the form of attribute associations, default values, value ranges of attributes, qualitative and quantitative relations which have been suggested being useful for guiding the design refinement in prototype-based design systems [12] and the design adaptation in case-based design systems [3]. The indexing mechanism of the system provides a user with flexibility in both design case retrieval and design prototype retrieval. The overall relationships between generalized design schemas, design cases and predefined design prototypes are shown in Figure 1.

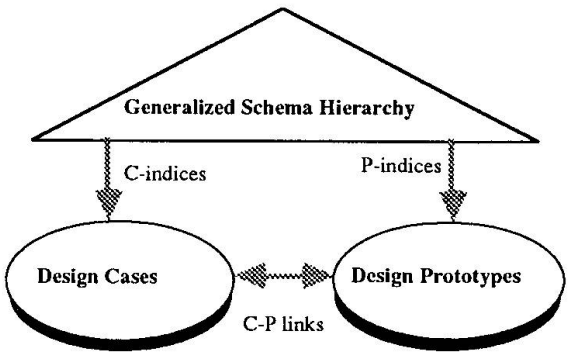


Figure 1. The relationships between generalized design schemas, design cases and predefined design prototypes. C-indices means case indices, P-indices means prototype indices, and C-P links means case-prototype links.

2. HETEROGENEOUS DESIGN KNOWLEDGE

In this section we discuss three types of design knowledge, predefined design prototypes, design cases, and computer-generalized design schemas.

2.1 Design prototypes

A design prototype is a conceptual schema for representing compiled design knowledge. The formal model of design prototype [2] is symbolically represented as follows:

$$P = (F, B, S, C, K)$$

where

P	a design prototype
F	a set of functional attributes
B	a set of behavioural attributes
S	a set of structural attributes
C	a set of contextual attributes
K	design knowledge

One representation of design prototype is frame-like. Each attribute may have type, unit, default value, and value range. For example, for attribute, horizontal-module of office buildings, the type is numeric, the unit is feet, the value range is 2 - 4, and the default value is 3.4. The types of attributes can be nominal, ordered, structured, numeric, set, and other design prototypes. Design knowledge can be if-then rules, qualitative and quantitative relations, and dependencies between variables.

A series of design prototypes, from specific to general, forms a design prototype memory. In a prototype-based design system, designing starts with function requirements from a client followed by the process of continually finding appropriate design prototypes using existing information as the index, deriving instances from them and synthesizing these instances to compose the desired artifact. The speed of prototype retrieval and the degree of relevance between new design problem and the retrieved design prototype have strong influences on the system efficiency.

2.2 Design Cases

A design case is represented as a set of features or attribute-value pairs which are classified into four categories of function, behaviour, structure and context. It is formally defined as follows:

$$\text{Design case} = (F, B, S, C)$$

where

F	a set of functional features
B	a set of behavioural features
S	a set of structural features
C	a set of contextual features

Like design prototypes, the type of an attribute here may be nominal, ordered, structured, numeric, set, and a particular design prototype.

2.3 Design schemas

The memory system will automatically generalize knowledge-rich design schemas from design cases. We call them knowledge-rich design schemas because they contain more information than usual generic design cases. A generalized design schema consists of common features, generalized features, generalized knowledge (qualitative and quantitative relations), and typology (indices to design cases, more specific schemas, more general schema, and related design prototypes) (Figure 2).

Generalized Design Schema:
- link to more general design schema:
- common features:
- generalized features:
- generalized knowledge:
  - qualitative relations:
  - quantitative relations:
- index or discrimination nets:
  - link to design cases
  - link to more specific design schemas:
  - link to related design prototypes

Figure 2. The representation of a generalized design schema

Common features are those shared by all cases under the schema and each of generalized features has a value range and often a default value. By quantitative relations here we mean empirical formulae among numeric attributes (e.g., for a given set of examples of office buildings, the system has learned that maintenance-cost-per-year is approximately equal to $0.1 \times$ total-cost-of-construction). Qualitative relations among numeric attributes are represented as empirical, proportional relations (e.g., for a given set of examples, number-of-building-stories$^+ \rightarrow$ cost-per-efficient-area$^+$; building-performance-level$^+ \rightarrow$ ratio-of-net-area-to-gross-area$^-$; the last relation says that the rise of building performance level may lead to the reduction of ratio of net area to gross area). These relations can be used as heuristics to guide the design adaptation in case-based reasoning and the instance refinement in prototype-based design, or aid a designer in making decision at preliminary design stage.

3. THE MEMORY ORGANIZATION

3.1 The General Strategy

In general, machine learning techniques are used to create knowledge-rich design schemas which function as both a knowledge representation schema for storing generic design cases and design knowledge learned from design cases, and a memory organization facility for linking design cases to related design prototypes as shown above in Figure 1.

The design schemas are further classified into F-schema, B-schema, S-schema and C-schema. F-schema is a schema indexed with function descriptions and its creation is dependent on a cluster of functional features; similarly, B-schema is a schema indexed with behaviour descriptions, S-schema is a schema indexed with structure descriptions, and C-schema is indexed with the description of design context. Each of classes of schemas forms separate hierarchies, and in a whole they construct an efficient memory that stores and retrieves design cases, generalizes more general knowledge from design cases, and associates relevant design cases, design schemas and design prototypes together (Figure 3).

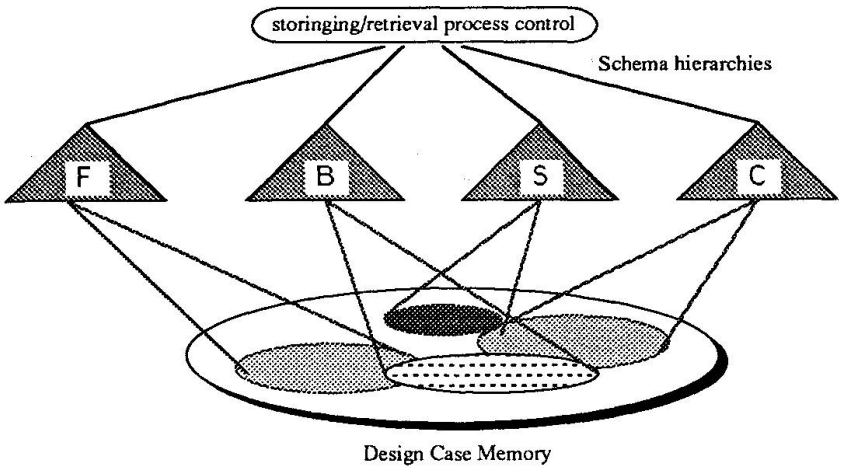


Figure 3. Design cases can be retrieved with each of four schema hierarchies of function, F, behaviour, B, structure, S, context, C, or any combination of them.

3.2 Main Processes

Two main processes of memory organization suggested in Wang and Gero [13] are conceptual schema hierarchy formation and numeric relation discovery, Figure 4. The techniques used in the former are influenced by the generalization-based memory techniques applied in the UNIMEM [5]; the techniques used in the latter are similar to those presented in ABACUS[1] and BACON [4]. The rest of this section will describe these processes.

Input —> Conceptual Schema Hierarchy Formation —> Numeric Relation Discovery —> Output
(Design Cases) (Schema Hierarchy)

Figure 4. Two main processes of the memory organization: conceptual schema hierarchy formation and numeric relation discovery.

Process of conceptual schema hierarchy formation

1. Receive new design case;
2. Let Feature-list = Functional attributes of new design case;
3. Let most-specific-schemas = results of SEARCH (root-of-F-hierarchy, Feature -list, design-case);
4. For each of nodes in the paths of search, call QUALITATIVE-DISCOVERY process;
5. For each of Most-specific-schemas, do

   5a. If the similarity between new design case and one indexed under the schema is higher than a predefined threshold,

   create a new schema,
   remove the case index from the original schema,
   create index from original schema to new schema,
   create index from new schema to the design cases,
   generalize default values and value ranges of attributes;

   If any feature in the schema has a prototype name as its value,
   create the index from the schema to that prototype,
   call a process CONSISTENCY-MAINTAIN(Design-case, prototype-name)
   to update the related prototype;

5b. If no generalization can be done,
add new case into the schema,
update the index to design cases
update the default values and value ranges of attributes.

The system incrementally incorporates design cases into a hierarchy by this process. Various indices will be created. The SEARCH process is similar to that in UNIMEM except that only the indexing of a schema and Feature-list will be considered in matching. The process QUALITATIVE-DISCOVERY will create or evaluate qualitative relations and may call a quantitative discovery process. Both processes are described below. The process CONSISTENCY-MAINTAIN will check related design prototype and may modify default value or value range of an attribute. B-schema hierarchy, S-schema hierarchy and C-schema hierarchy are created and updated with the same process with minor changes as in steps 2 and 3. Part of a generalized schema hierarchy and its relations to design case and design prototypes is graphically described in Figure 5.

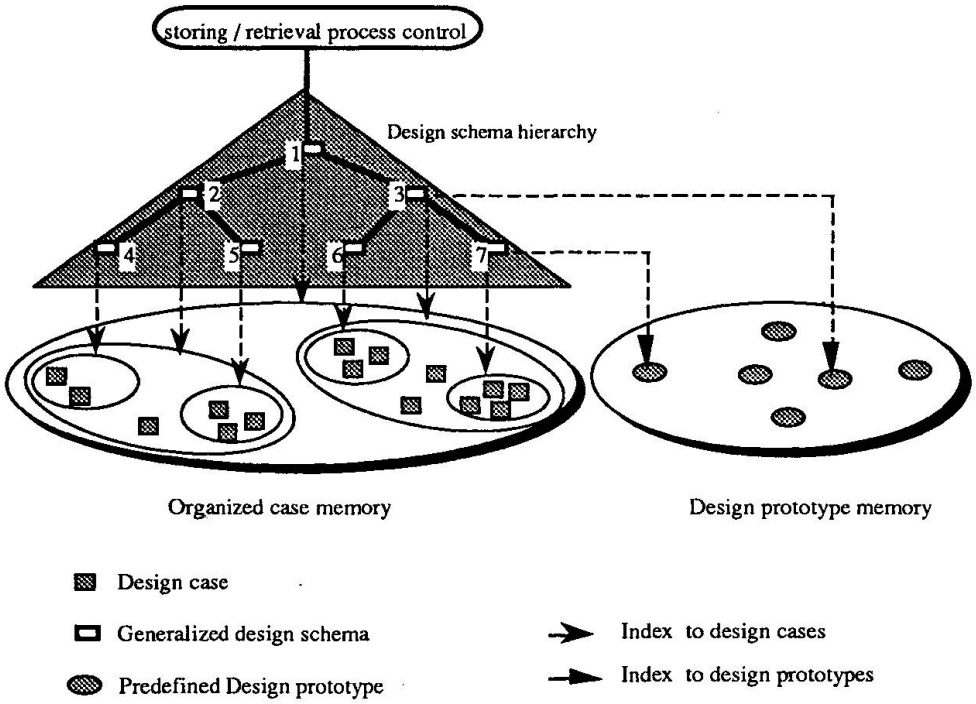


Figure 5. Part of detailed representation of generalized schema hierarchy and its relations to design cases and design prototypes.

The numeric relation discovery itself consists of two processes: qualitative relation discovery QUALITATIVE-DISCOVERY and quantitative relation discovery QUANTITATIVE-DISCOVERY. Quantitative relation between two attributes are discovered by the following process:

Process of qualitative relation discovery between X and Y

1. As design cases are sequentially incorporated into the memory,
check the directions of their value changes in last two cases;
if both go in the same direction (increase or decrease),
let counter P increase by 1;
if one variable increases and another decreases,

let counter N increase by 1,

2. If P / (P + N) > predefined threshold T1,
   hypothesize a relation $X^+ \rightarrow Y^+$;
3. If N / (P + N) > threshold T1,
   hypothesize a relation $X^+ \rightarrow Y^-$;
4. If a hypothesis has already created,
   both P / (P + N) and N / (P + N) < predefined threshold T2,
   propagate the hypothesis into sub-schemas except one that current case belongs to,
   remove the hypothesis from the original schema;
5. If P + N > predefined threshold T3,
   call a process QUANTITATIVE-DISCOVERY.

The value of P/(P+N) or N/(P+N) is regarded as the confidence of hypothesis. When it becomes smaller than the predefined threshold T2, the hypothetical relation is disproved and deleted; on the other hand, when the number of cases (P+N) becomes larger than predefined threshold T3, quantitative discovery process QUANTITATIVE-DISCOVERY will be called, which is described as follows.

Process of qualitative relation discovery between X and Y

1. If $X^+ \rightarrow Y^+$,
   create new variable X/Y, let C equal to the value of X/Y;
   if C remains constant or its deviation is smaller than predefined threshold T4,
   hypothesize an equation X/Y = C;
2. If $X^+ \rightarrow Y^-$,
   create new variable X×Y, let C equal to the value of X×Y;
   if C remains constant or its deviation is smaller than predefined threshold T4,
   hypothesize an equation X×Y = C;
3. If a hypothesis has already been created,
   update the deviation of C;
   if the deviation of C becomes larger than the predefined threshold T4,
   delete the hypothesis.

As described in the above processes, for each of schemas which contains more than two design cases, the system starts searching for qualitative relations, while the search for quantitative relation will be carried out only if a qualitative relation has been found and the number of supporting cases has become equal to or larger than a pre-defined threshold. Since the generalized relations are assumed as a kind of heuristics, not as scientific laws, they may be approximate but still useful. The computational cost of relation discovery is high. To control the combinatorial explosion, the system may only search the potential relations between those variables which have dependency relations with each other. Design prototypes provide such information. The underlying assumption is that it is relatively easy for experts to tell if there exists a dependency between two variables but it is much more difficult for them to describe mathematically the relations between them.

As we have seen from the above processes, once a relation is hypothesized, it can be incrementally evaluated as new design cases come. If a quantitative relation at a high-level schema does not cover a new case, the system will not simply throw it away, but propagate it into sub-nodes which the new case does not belong to (in effect it just reduces the application scope) and search for a qualitative relation at original node.

Process of memory consistency maintenance

When a schema is created or updated, if there is an index from a schema to a design prototype, the related design prototype(s) will be checked by the process of CONSISTENCY-MAINTAIN. The value range

of a variable in design prototypes will be expanded if it does not cover the range of the variable in the generalized schema. This gives the memory the capability of self-adaptation.

4. RELATED WORK

Using machine learning techniques to organize case memory and acquire knowledge from design cases was investigated by Reich et al. [9]. In their system, Bridger, only two types of knowledge were considered, design cases and generalized design schemas. Neither qualitative nor quantitative relations are generalized in the schemas. However, it is possible to insert qualitative and quantitative discovery processes in the process of concept formation and generalize more knowledge in the schemas. Design prototypes may also be integrated into the system. The combined application of concept learning and quantitative discovery in design was also examined by Maher and Li [7, 6]. Although a similar two-stage strategy was applied in their system, they utilized a probabilistic association network and classical numerical methods, regression analysis and dimensional analysis, and approached concept learning and quantitative discovery in a non-incremental fashion. Their system, unlike our model presented above, only focused on automatic design knowledge acquisition from design cases and neither indexing nor pre-defined design prototypes needed to be considered. Case memory organization and combined application of design cases and qualitative models was explored by Goel et al. [3]. In their system, Archie, design cases are indexed by a different categories of features, such as design specifications, design solutions, design outcomes, design critiques, potential problems, and adaptation strategies.

5. CONCLUSION

We have presented a machine learning approach to the memory organization for heterogeneous design knowledge in form of design cases, design prototypes and design schemas. The integrated application of concept formation and qualitative and quantitative discovery techniques in design is original. To combine design cases and design prototypes for more powerful knowledge-based design systems, their memory organization needs to be further explored. The efficiency of qualitative and quantitative relation discovery in an incremental fashion needs to be empirically analysed. What should be in a design case and what should be in a design prototype need to be further justified in the area of their integrated applications.

REFERENCES

1. Falkenhainer, B. and Michalski, R. S. (1986). Integrating quantitative and qualitative discovery: the ABACUS system, *Machine Learning* **1**: 367-402.
2. Gero, J. S. (1990). Design prototypes: a knowledge representation schema for design, *AI Magazine* **11**(4): 27-36.
3. Goel, A. K., Kolodner, J. L., Pearce, M., Billington, R. and Zimring, C. (1991). Towards a case-based tool for aiding conceptual design problem solving, *Proceedings Case-Based Reasoning Workshop*, Washington, DC.
4. Langley, P., Simon, H. A., Bradshaw, G. L. and Zytkow, J. M. (1987). *Scientific Discovery*, MIT Press, Cambridge.
5. Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM, *Machine Learning*, 2: 103-138.
6. Maher, M.L. (1992). Automated knowledge acquisition of preliminary design concepts, *ASCE 8th Conference on Computing in Civil Engineering*, Dallas, Texas, pp. 975-982.
7. Maher, M. L. and Li, H. (1992). Automatically learning preliminary design knowledge from design examples, *Microcomputers in Civil Engineering* 7: 73-80.

8. Maher, M. L. and Zhang, D.M. (1991). CADSYN: using case and decomposition knowledge for design synthesis, *in* J. S. Gero (ed.) *Artificial Intelligence in Design*, Butterworth-Heinemann, Oxford, pp. 137-150.
9. Reich, Y. and Fenves S. J. (1991). The formation and use of abstract concepts in design, *in* D. H. Fisher, M. J. Passani and P. Langley (eds), *Computational Approaches to Concept Formation*, Morgan Kaufmann, San Mateo, CA.
10. Rosenman, M. A., Gero, J. S. and Oxman, R. E. (1991). What's in a case: the use of case bases, knowledge bases and databases in design, *in* G. N. Schmitt (ed.), *CAAD Futures'91*, ETH, Zurich, pp.263-277.
11. Sun, D. and Gero, J. S. (1991). Representing design experience through design cases: the use of case-based reasoning in design, *in* G. Woodbury (ed.), *The Technology of Design*, ANZAScA, University of Adelaide, Adelaide, pp.235-244.
12. Tham, K. W. (1991). A model of routine design using design prototypes, *PhD Thesis*, Department of Architectural and Design Science, University of Sydney, Sydney.
13. Wang, W. and Gero, S. J. (1991). A model for the organisation of design prototype and design case memories, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design, Preprints of the Workshop of IJCAI-1991, Sydney*, University of Sydney, Australia, pp.301-307.