Zeitschrift: IABSE reports = Rapports AIPC = IVBH Berichte

Band: 58 (1989)

Rubrik: Session 4: Expert systems for codes and in other areas of civil engineering

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Mehr erfahren

Conditions d'utilisation

L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. En savoir plus

Terms of use

The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. Find out more

Download PDF: 11.02.2026

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

SESSION 4

Expert Systems for Codes and in other Areas of Civil Engineering

Application de systèmes experts dans les normes et dans d'autres domaines du génie civil

Expertensysteme für Normen und andere Gebiete des Bauingenieurwesens

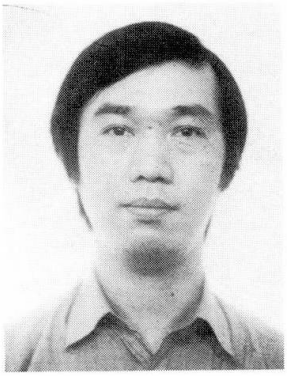
Leere Seite
Blank page
Page vide

Expert System for Tunnel Design and Tunnelling

Système expert pour la conception des tunnels

Expertensystem für den Entwurf von Tunneln

Hajime OSAKA
System Engineer
Taisei Corporation
Tokyo, Japan



Hajime Osaka, born in 1947, obtained his degree of M.Sc. in Mathematics at the Tokyo Institute of Technology. For eighteen years, he has been actively involved in the fields of Geomechanics and Numerical Analysis. He, now in a general construction firm, is a senior administrator of information systems department.

T. Shinokawa, Sato Kogyo Corp., Japan
Y. Goto, Tokyu Construction Corp., Japan
T. Tsuruhara, Oyo Corp., Japan
Ö Aydan, Nagoya Univ., Japan
Y. Ichikawa, Nagoya Univ., Japan

SUMMARY

In this article, we describe an expert system for tunnel design, its structure and its features, and present some applications of the system. The system consists of four sub-expert systems (1 - ES for Standard Tunnel Design Methods, 2 - ES for Framed Structure Method, 3 - ES for Theoretical Design Methods, 4 - ES for Numerical Analysis Design Method), and a common part to control the overall system and two data-base systems (1 - Tunnel Data - Base System, 2 - Rock Mass Data - Base System).

RESUME

L'article présente le système expert pour la conception des tunnels ainsi que sa structure et ses caractéristiques, ainsi que quelques applications. Ce système se compose de quatre sous-systèmes experts 1: Sous-système expert pour la méthode standard de conception des tunnels; 2: Sous-système expert pour la méthode de structure charpenteé; 3: Sous-système expert pour les méthodes de conception théorique; 4: Sous-système expert pour la méthode de conception par analyse numérique. Le système comprend aussi une partie commune contrôlant le système ainsi que deux systèmes de bases de données 1: Système de base de données pour les tunnels; 2: Système de base de données pour des roches.

ZUSAMMENFASSUNG

Im vorliegenden Aufsatz beschreiben wir unser Expertensystem für den Entwurf von Tunneln und führen einige Anwendungsbeispiele auf. Es besteht aus vier Expertenteilsystemen (1. Standardmethoden des Tunnelentwurfs, 2. Rahmentragwerksverfahren, 3. Theoretische Methoden des Tunnelentwurfs, 4. Numerische Analysemethoden des Tunnelentwurfs), einem gemeinsamen Teil für die Kontrolle des Gesamtsystems und zwei Datenbanksystemen (1. Tunnelbau-Datenbanksystem, 2. Gebirgsmassiv-Datenbanksystem).

1. INTRODUCTION

The design of geotechnical engineering structures generally involves many elements of experiences. The reason for this is due to the difficulty of evaluating the true mechanical behaviour of ground at the stage of designing. This is well-pronounced in the case of tunnel design. The design in many cases is carried out with insufficient information on the geology and the mechanical behaviour of the ground and through some simplifications regarding the geological structure and the mechanical modelling of the ground on the basis of experiences of specialists. The decisions differ from one to another depending upon the purpose of tunnelling, the objective of designing, the accuracy of available input data, the effect of designing on construction procedures and social constraints. In addition, the experiences of the designers influence the decisions up to a great extent. To use past experiences on a tunnel, the investigation of a number of items are usually necessary and even sorting out the investigated items present a great amount of work and difficulty.

As expert systems (called ES hereafter) have become popular in recent years, we have started to investigate how to systemize the experiences and decision making procedures of experts in tunnel design. In our study, we are mainly concerned with tunnel construction procedures by the New Austrian Tunnelling Method (NATM), since the NATM is the most widely used tunnelling technique in Japan.

The authors have constituted a joint research group under the leadership of Nagoya University, involved with the tunnel design, and their experiences are systemized for the development of Expert System (ES) for the design and construction of tunnels (the project for developing the tunnel expert system: TUX project). The present work has been carried since April 1986 till March 1989 and we herein present some outcomes of our work up to now. The joint research group have been consisted of 20 people closely involved with the design of tunnels and the work has been carried out with the close colloboration of the members. Firstly, the steps and elements of design procedures were carefully investigated and, on the basis of this investigation, the levels of main steps of design procedures were then defined. As a result of these studies, the expert system (ES), consisting of four sub-expert systems has been developed. The sub-expert systems are; Expert system for the tunnel design standards; Expert system for the analytic tunnel design methods; Expert system for the design of tunnel supports by the framed structure method; Expert system for the numerical analysis. At the same time, tunnel and support data-base systems and rock tests data-base system have been developed using micro-computers as it was concluded that it would be necessary to accumulate and store the experiences with the ES.

2. SELECTION OF THEMES OF DEVELOPMENT

The present expert systems are concerned with functioning either as a specialist and/or dealing with uncertainities. Though the functioning of the system as a specialist involves some kind of uncertainity, the systems can be usually classified to one of the classes depending upon their objective. Our expert system is concerned with functioning as a specialist.

Tunnel design concept has undergone a great transformation with the introduction of the New Austrian Tunnelling Method (NATM) and the construction equipments and procedures have been renewed as a result. The main principle of the NATM is associated with the effective use of the circuit of Investigation-Design-Construction steps as compared with the conventional design. The principles of the NATM are very logical and it incorporates the experiences of engineers in the all steps of the tunnel construction. Japan Society of Civil Engineers (JSCE) has designated the NATM as the standard tunnel design & construction method and tunnel constructions by the NATM are expected to increase more and more. For the further development

of the NATM as more economical and rational method, it is necessary to develop more effective numerical and theoretical analysis methods, optimum control values, to establish design alteration procedures, to check the suitability of the employed design and to accumulate the experiences. Therefore, a data gathering and a unified NATM design and construction system is considered to be necessary. This system should have mechanism to incorporate not only the experiences but also new technical developments. The ES based on the artificial intelligence concept is thought to be suitable for such a purpose.

The research group first analysed the tunnel design items and the associated procedures. The tunnel design can be classified into two stages; the design before construction (initial design) and the design during construction. It is a problem to design tunnels rationally on the basis of little amount of geological and experimental data. However, it is possible to do more accurate designs with the information gained from the performance of the tunnel and the observation of the face. In the design during construction, though the procedure is also the same as that in the initial design, the interpretation of measurements and the observation of the face are included. The present work is mainly concerned with the initial design. Fig. 1 shows the NATM design flow chart, which is concluded from the analysis of the initial design procedures.

The features of each procedures and the sistemized items are as follows.

(1) Data input

The first step is sorting and checking information for design which are defined as input data. The input data used in the tunnel design generally involve design conditions such as dimensions and the geometry of the tunnel, environmental constraints, geological decisions from geologic and past-record surveying, boring and core testing and ground classifications based on elastic wave velocity measurements. As the amount of data is too large and necessitate a great deal of labourship, the necessary data items are only included in the system in association with the capability of the present system.

(2) Determination of design method

Depending upon the ground conditions and the scale of the tunnel, the operation to determine the suitable design method from the input data is carried out. Presently this operation is left to the designer. The operation involves highly expertise knowledge and it is simplified in the present system.

(3) Determination of standard support pattern

The determination of the standard support pattern is concerned with the methods based on case studies and ground classifications. The determination of the support pattern from experimental studies during driving exploration adits or test adits is excluded in the system as this is involved with the concept of the design during construction. The ground classifications for the determination of the standard support patterns are installed in the system. The presently built-in classifications are the classification of Japan Roadway Association for roadway tunnels and the classification of Japan Railways for railway tunnels. The determination of the support pattern by the case studies is based upon the tunnelling data base system. The tunnelling data base system stores records of past tunnel constructions and the search are carried out through key-words and the support pattern is selected for the problem handled. The items have been still sorted. In addition, the examination of the counter measures for seepage and face instability are considered to be carried out at this level.

(4) Evaluation

The content of the evaluation is very large and there are various alternatives. More specifically, Some of these alternatives are as follows: 1-) A more detailed examination of items of the chosen standard support pattern, 2-) Decisions regarding the necessity to alter the chosen support pattern or the choice of the design method following the stability analysis, 3-) Exam-

ination of any item of input data or information is lacking or not. This part of the system is closely associated with the decisions based on the experiences of specialists. As the scope is too large to systemize, the present version of the TUX system covers the decision if there is any need to examine the chosen standard pattern and the selection of the stability analysis method.

(5) Stability analysis

The methods for stability analyses are well-established and programmed. This part of the work is associated with data preparation and the experience to evaluate the calculated results. Stability analysis methods consist of i-) Stress analysis of support members subjected to loosening loads, ii-) Limit equilibrium analysis of the bearing capacity of rock arch, iii-) Closed form solutions for the stability of ground and support members, iv-) Model tests, and v-) Numerical methods (FEM,BEM). In the present version of the TUX system, the model test method is excluded.

(6) Detailed Design

This operation involves the detailed design of elements following the primary support system. The elements in these category are the design of concrete linings, of portals, of waterproofing and drainage. The examination of construction methods, setting the control values and the evaluation of environmental effects are necessary in this stage.

As noted in Fig. 1, there is no arrows among the operation stages to indicate a certain flow path. This is due to the reason that the system should be flexible to incorporate several alternative paths effectively depending upon the conditions of each tunnelling problem handled. This is the usual procedure in experience-based designs. It is difficult to handle with various paths by the use of ordinary methods. On the other hand, this type problems can be easily dealt with the ES easily.

The development environment used in building the ES is the VAX_AI Station of the Digital Equipment Corporation (DEC) and the tool for the ES is OPS5. The computations are done using the existing but renewed programs written in FORTRAN.

3. SYSTEM OUTLINE

The structure of this tunnel design ES (TUX) is shown in Fig. 2. The TUX consists of a common & control section and several sub expert systems; i-) ES for standard design methods, based on ground classifications, ii-) ES for theoretical design methods, based on closed form solutions, iii-) ES for framed structure design method, based on the framed structure analysis of the support members and loosening load concept, iv-) ES for numerical analysis design method, based on the numerical analysis by finite element method (FEM). The TUX is presently built for the use in Japan and the presentations of conversations between users and the knowledge bases and the rules are all in Japanese.

3.1 Control and Commonly-referred Section

The control section of the TUX is to control the rules to execute each appropriate design method and to enable input of arbitrary tunnel shape and support pattern. The commonly-referred section consists of a knowledge base to determine material properties of ground and support members. In the judgement of ground conditions, the conditions such as expansionability or flowability of surrounding rock mass etc. are evaluated. The definitions of tunnel dimensions are shown in Fig. 4 and patterns of support systems are given in Table 1. However, the support system can be also input besides the support patterns chosen from the classifications. As for ground properties, an estimation procedure for the material properties of support members is available and Table 2 shows the units of the knowledge base. Each unit of the knowledge base is composed of several rules, and if they are called from other knowledge bases,

they send back results after applying the rules for the input information at the time of calling. And it checks whether the required data is available or not. If the data is not available, then it seeks for a knowledge base, in which the the required data is possible to be found. Then, the knowledge base is again applied to the data. A simple example is shown in Fig. 3. The knowledge base for the determination of the ground deformation modulus consists of a sequence of knowledges regarding the procedure how to determine the deformation modulus. As a specific example, the modulus is determined from the modulus of rock element and jointing index. If that rule is executed, then the corresponding knowledge base is called. The called knowledge base first checks whether the required data exists or not, if not found, then it calls the knowledge base, in which it can be found. Once the modulus of rock element and jointing index are available, then it evaluates the modulus of ground and send it back to the knowledge base which has required that information. The smallest unit of knowledge bases corresponds to the knowledge of experts for each theme. In this case, the definition of the formulae to determine the ground modulus from the modulus of rock element and jointing index is a knowledge base together with the limitations of these formulae. The ES is evolving by accumulating this type of contents.

3.2 ES for Standard Design Methods

The classifications of Japan Railways and Japan Roadway Association are the standard design methods for tunnels in the respective tunnel types. A sub-expert system has been developed for the tunnel design, based on the ground classifications and the standard support patterns for each respective class, developed by the above associations. The system consists of ground classifications and the determination of the tunnel shape and support pattern corresponding to each ground class. These classifications are outcomes of numerous actual tunnel construction practices. The name of rocks, their elastic wave velocity and the ratio of strength to overburden stress are fundamental data for the stability analysis and the TUX provides a data-base of the existing tunnels. A knowledge base is installed for searching the data-base and treating exceptional cases. There are several tunnel standard cross-sections provided by each respective authority, which are the elements of the knowledge base for the dimensions of tunnel shapes as shown in Fig. 4. The support system is determined from the dimensions of the tunnel cross-section and rock classes together with the information on excavation conditions and the installation patterns of support members. These relations are all installed in the system as a knowledge base.

3.3 ES for Theoretical Design Methods

There are several simple closed form solutions suggested for the tunnel designs by NATM. Of these, the methods suggested by Einstein, Egger and Oka are installed in this sub-expert system. The TUX first selects the calculation method from the input conditions given in Table 3. Then, the modelling of ground, in which the strength and deformation properties of ground are determined by using the determination function of the commonly-referred section for ground properties, is done automatically as britte, perfectly-plastic or strain-softening plastic type. The tunnel shape is approximated as a circle and the support pattern is modified for that shape. For the material properties of support members, the commonly-referred section is used. Once the data are ready, the stress state, tunnel wall displacements and the plastic zone radius are calculated. Then, the check on the appropriateness of the support pattern is carried out by comparing the resistances of support members with the calculated results. The comparison method is shown in Fig. 5.

3.4 ES for Framed Structure Design Method

As a design method of the support system, a framed structure modelling is employed, in which loads are determined from loosening load concept. This sub-ES is coded as shown in Fig. 2. In the system, "the construction state" involves the data for the excavation steps and the hardening state of shotcrete in relation to the tunnel face advance. In setting the calculation case, the conditions for the installation timing of the support members, the excavation method and the presence of the primary and secondary linings are determined. The condition for the element divisions of support members, their material properties and boundary conditions are set. To determine the ground reaction coefficient, the system provides a knowledge base with various experimental data. The loads to act on the support members are determined from the Classification of Terzaghi in relation with the Classification of the Central Research Institute of Electric Power Industry of Japan. The calculated results are the displacement, the axial and shear forces and the moment of support members. An example is given in Fig. 6. From these figures, the stress intensity and displacement of the members are checked.

3.5 ES for Finite Element Analysis (FEM)

For more complex problems, numerical analysis methods are used for the stability analysis. In the TUX, a general finite element program is installed. This program treats the rock mass as a elastic medium under plain-strain condition and takes into account the effect of the face advancing. The face advance conditions involve the variations of support patterns and the material properties of the support members in relation with the initial-stress release rate. When the overburden is shallow and/or adjacent structures are present or the opening is very large, the numerical analysis methods are usually used. In the TUX, the mesh generation, the modelling of ground and support members, the setting of the analysis domain and the presentation of the calculated results are all done automatically. The analysed cases are the same as those in the case of the framed structure ES. The setting of the analysis domain is done with a minimum number of elements while taking the care of accuracy of the calculations. When the overburden is shallow, the inclination of the surface is also taken into account. The material properties of ground and support members are determined from the knowledge base of the commonly-referred section of the system. In determining the initial stress state, the value of the poisson ratio is assigned by using a rule of experience. A knowledge base is available for the stress release rate function, which is determined from 3-D FEM analyses, for simulating the effect of the face advancing. The ground, rockbolts, steel ribs, shotcrete and concrete lining are represented by 4-noded isoparametric elements, line elements and 4-noded isoparametric elements with the use of the reduced integration technique respectively. The calculation scheme models the simulation of the face advance and excavation states and the calculated results are displayed and/or output as the distributions of displacements, stress in ground and support members and safety factor. Fig. 8 shows the examples of the distributions of safety factor contours after the excavation of the upper and lower-half sections respectively.

It is difficult to write the rules of experiences for the evaluation of the results of the finite element analysis as the evaluation is generally done after seeing the distributions of stress and displacement and safety factor contours. Presently, the evaluation scheme for the finite element calculations is still being installed in the TUX. The results shown in Fig. 8 are obtained after 30 minutes using the TUX.

4. Evaluation of the ES

The TUX has the function of carrying out an automatized design as it generates the necessary data for the tunnel design from the knowledge of experiences. To activate the TUX, the minimum amount of data are; the purpose of use (roadways, railways, etc.), rock name, elastic wave velocity of ground, the type of the standard tunnel cross-section, overburden, inclination of the ground surface and the pull-out capacity of each bolt. If these data are supplied, then the TUX deduces the tunnel shape, and the support pattern and carries out the stability check analysis. Still some problems exist for inexperienced users in regard with the present automized system. The current system can be regarded as an additional support tool for the experienced designers and can be used as a tool for the education of inexperienced users under the supervision of the experts.

During the development of the TUX, we have understood the true features of expert systems; some parts were well-established while the others were needed to be revised and re-written. We state our opinions about these as follows. The merits of expert systems of the TUX type can be said to be:

(1) Deduction Ability

As the ES has its own deduction system to deal with the rules of experience, there is no need to prepare another special routines for this purpose. This feature is particularly important in the application when no fixed procedure is available for the problem.

(2) Object-orientated structure

The ability of the detailed checking of the small elements of the knowledge bases is more powerfull than that by conventional programming techniques. This, in turn, enables to easily code and check the program in segments separately.

(3) Easy understanding

The description of rules is close that in human language so that it is easy understand once one knows how to read.

(4) Certainty factor and fuzzy theory

It is possible to incorporate the decision making procedure by the use of the certainty factor concept or the fuzzy theory. The certainty factor and fuzzy have been introduced by Imazu[4] and Shimizu[5] for the rock classification expert system respectively. These approaches can be easily introduced for the determination of properties of ground and the evaluation of the calculated results.

As it is clear from the above statements, the easiness of extending the system is the most important feature of the ES. This becomes particularly important in further development of the large scale expert systems such as the extended TUX involving the management and measurements in tunnelling as well.

A revision of the TUX in future is considered to be closely dependent on the development of the tunnel data-base system. We have presently developed data-base systems for tunnels and rock properties, using the personal computers, which are independent of the TUX.

The tunnel data-base system has been developed with the objective of checking the design with reference to experiences gained in the existing tunnels under the similar ground and environmental conditions. Table 4 gives the items of the data-base system. The rock poperty data-base system has been developed with the purpose to refer the properties of ground. The items of the data-base system are given in Table 5. Presently, the data gathered from 96 tunnels in Japan has been stored in the tunnel data-base system. While the rock property data-base system has 1300 data. The data gathering process has been still continued. How to use these data-base systems in connection with the TUX has been presently discussed and remains as a problem to be solved.

5. CONCLUSIONS

We have developed an automatized tunnel design system (called the TUX - Tunnel Expert System), based on the ES concept. It can be used by anyone hwo does not have sufficient knowledges of carrying out theoretical, framed structure or numerical analysis. Off course, there are several approaches how to position the ES in civil engineering. One of those is that the ES can be considered to act as an effective connection tool among Design - Construction - Research (Fig. 8). By starting to develop an expert system for the purpose of design and construction, we have been able to sort out the rules of experience up to this extent. By the application of the system to actual examples, more research themes will appear. In addition, more research topics are expected to come into being at the time of coding the gathered actual examples in the ES. By coding the outcomes of these research topics in the ES will reflect themselves in design and construction. Nevertheless, this is only possible with the long-term contribution of numerous researchers and engineers. It is expected that the ES with the reflected outcomes in the above cycle will make themselves to function more smoothly and effectively.

It is a pity that the present system has not reached the above ideal state yet. This is partly due to the ability of the team members and partly due to insufficient development environment. Particularly, the problem regarding the cost and the user-interface of the ES is great in the development of the system with the help of a numerous engineers. Nevertheless, we expect that this problem will be overcome quickly in the near future.

Acknowledgements

The present research work has been jointly carried out from April, 1986 to March, 1989 by Nagoya University and Private Construction Companies under the Grant in Aids for Developmental Scientific Research, the Japan Ministry of Education (Monbusho). The authors wish to extend their sincere thanks and gratitudes to Prof. Dr. T. Kawamoto, Nagoya University for his supervision and the members of the research team for their continuous contributions and the Japan Digital Equipments Corporation (DEC) for their co-operation and help.

References

1- JSCE ed. : Tunnel standards manual (Mountain tunnels), Tunnelling Commitee of Japan Civil Engineers (JSCE), 1986.
2- Japan Roadway Association: Tunnels in Design Manual, Chapter 9, Vol. 3, 1985.
3- Japan Railways (JR): NATM Design & Construction Manual, Japan Railway Construction Commitee, 1983.
4- Imazu, M. : Rock classification by using expert system. Procs. of the 19th Rock Mechanics Symposium, 1987.
5- Shimizu, N. : A research on rock classification by using fuzzy theory. Procs. of JSCE, No. 370/III.
6- Kawamoto, T. ed.: An expert system for tunnel design and construction. Research Rep., Project No. 61302060, Grant in Aids for Developmental Scientific Research, Japan Ministry of Education (Monbusho), 1989 (under preparation).

Table 1: Data for support pattern

Class	Name of data item
Excavation Conditions	Excavation method Advance ; Length of round
Rockbolt pattern 1	Rockbolt installation type Rockbolt length Rockbolt number Rockbolt diameter Rockbolt transverse spacing Rockbolt longitudinal spacing Pull-out capacity
Rockbolt pattern 2 (Two type rockbolt installation patterns are possible)	Rockbolt installation type Rockbolt length Rockbolt number Rockbolt diameter Rockbolt transverse spacing Rockbolt longitudinal spacing Pull-out capacity
Shotcrete	Definition method for shotcrete thickness Shotcrete thickness (Upper) Shotcrete thickness (Lower)
Steel ribs	Steel rib type Area of the cross section (Upper) Area of the cross section (Lower) Installation spacing
Concrete Lining	Thickness of arch and wall sections Thickness of invert
Allowed displacement	Allowed displacement (Upper) Allowed displacement (Lower) Allowed displacement (invert)

Table 2: Knowledge base for determing ground properties

Item	Data to be determined	Determination method
Ground Properties	Unit weight	Input of test value
	Longitudinal velocity V_p	Input of measured value
	Shear velocity V_s	Input of measured value
	Ground strength	Input of test value
		Modification of rock strength by jointing index
		Determination from CRIEPI classification
	Initial stress	Overburden x Unit weight ($\gamma \times H$)
	Lateral stress coefficient	Input of measured value
		Estimation from Poisson's ratio
	Deformation modulus	Input of measured value
		Modification of elastic modulus of rock by jointing index
		Modification of dynamic elastic by Kujundzic's method
		Estimation from CRIEPI classification
	Poisson's ratio	Input of measured value
		Use of dynamic poisson's ratio
		Estimation from Ikeda's classification
		0.3
	CRIEPI's class	Input of user decision
		Estimation from wave velocity
	Ikeda's class	Input of user decision
		Rock name and wave velocity
Rock Properties	Strength	Input of test value
	Deformation modulus	Input of test value
	Wave velocity	Input of measured value

Table 3: Setting of design cases

Initial stress	Behaviour	Calculation method
Non-hydrostatic	Elastic	Einstein
Hydrostatic	Elastic-perfectly plastic	Einstein
		Egger
	Elastic-strain softening plastic	Egger
		Oka
	Elastic-brittle plastic	Egger

Table 4: Items of tunnel data-base system

Content	Item
Tunnel	tunnel number, tunnel name, location, tunnel type, tunnel length, length by NATM, existence of adjacent constructions, existence of junctions, dates of commencement and completion of tunnel, referred article
Geologic Conditions	geologic age, geologic class, upper & lower elastic wave velocities, rock name, rock classification and classes, max. amount of seepage seepage period, expansibility, flowability, unusual loads, existence of faults & fractured zones, existence of landslides, overburden
Design Conditions	excavation methods, excavation types, diameter of excavation, excavation area excavation advance length, rockbolt (length, number, spacing)
	steel ribs (upper-half, lower-half, spacing), shotcrete (thickness, strength), existence of Bernold sheets, existence of steel-fibre, concrete lining (invert, arch, side-walls)
Instability	contents of instability, auxilary methods
Monitoring	max. displacement, max. crown convergence, pull-out resistance of bolts

Table 5: Items of rock-mass data-base system

Content	Item
Rock	Construction name, location, geologic age, formation name, rock name
Geologic Conditions	classification name and classes, V_p & V_s of rock mass, V_p & V_s of rock
Test data	excavation method, support pattern name, excavation area, unit weight, compressive strength, elastic modulus (E_{50}), Poisson's ratio, friction angle, cohesion, N-value, RQD, referred article

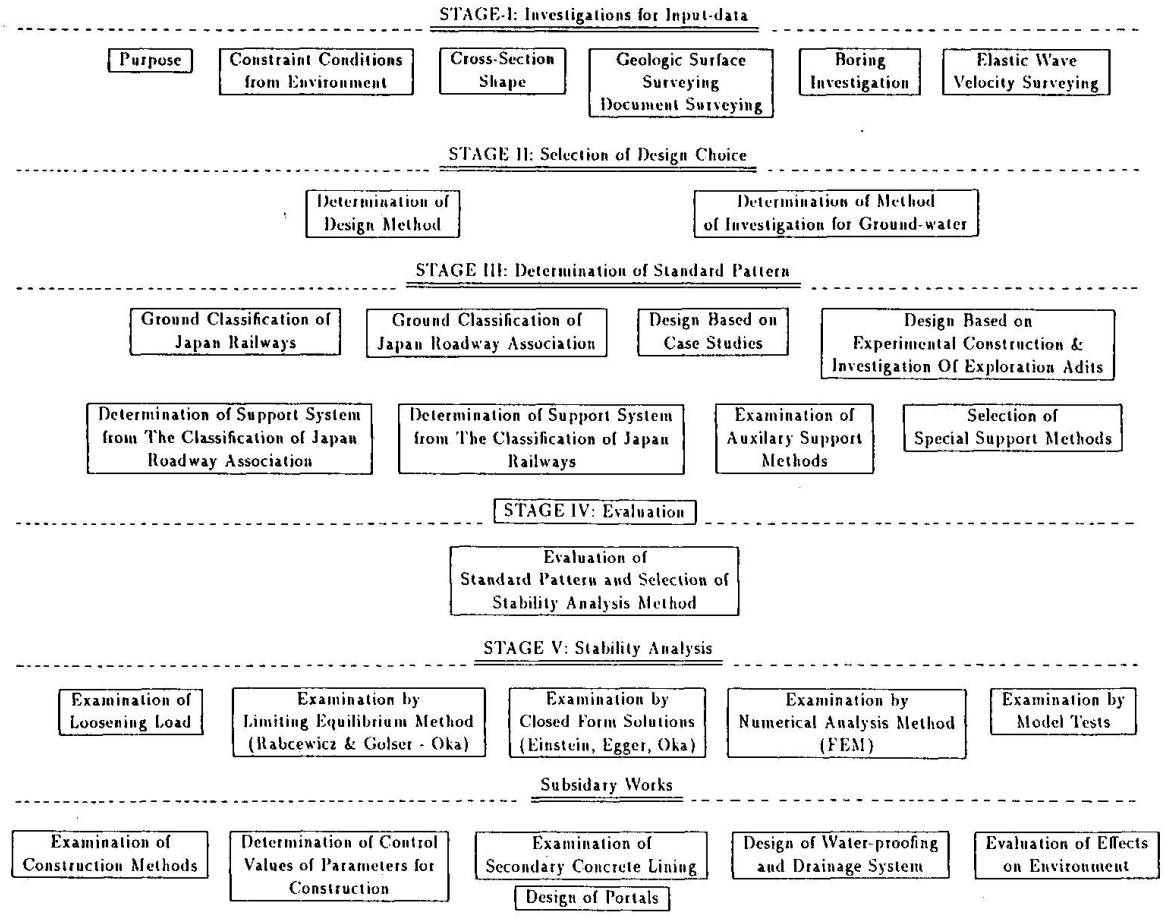


Fig. 1: The flow chart of the tunnel design procedure by the NATM

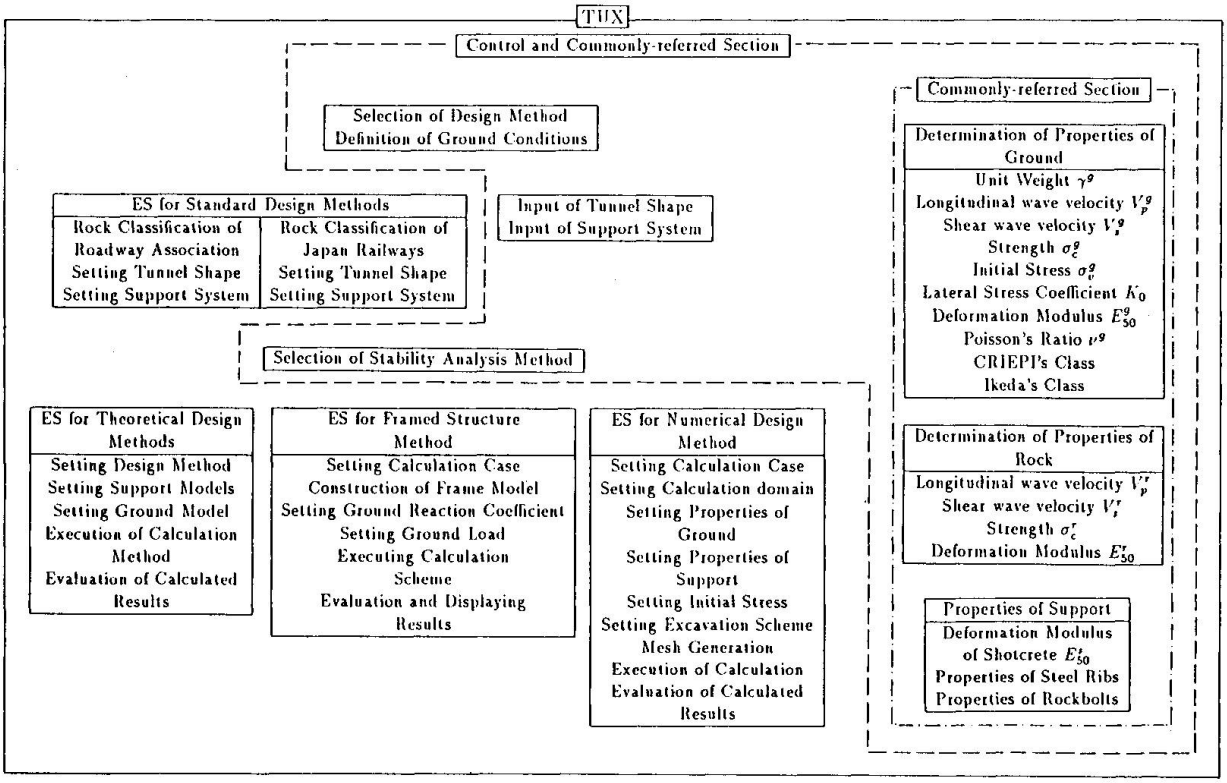


Fig. 2: The outline of the tunnel design expert system - TUX

Knowledge base for determining the deformation modulus of ground	
IF	the determination of the elastic modulus of ground is not carried out by using the jointing index,
THEN	call the knowledge base for calculation by using the jointing index

Auxilary knowledge base for determining the deformation modulus of ground from the deformation modulus of rock, by using the jointing index coefficient	
IF	the elastic modulus of rock is not known,
THEN	call the input knowledge base
IF	the wave velocity of ground is not known,
THEN	call the input knowledge base
IF	the wave velocity of rock is not known,
THEN	call the input knowledge base
IF	the elastic modulus of rock, wave velocity of ground and rock exist
THEN	calculate the modulus from $J_v = (V_p^g/V_p^r)^2$ $E_{50}^g = J_v \times E_{50}^r$
IF	the elastic modulus of rock, wave velocity of ground and rock do not exist,
THEN	there is no jointing index value

Input knowledge base for the deformation modulus of rock	
IF	the uniaxial strength of rock exists
THEN	get the elastic modulus of rock from uniaxial compression test results

Fig. 3: Calling scheme of knowledge bases

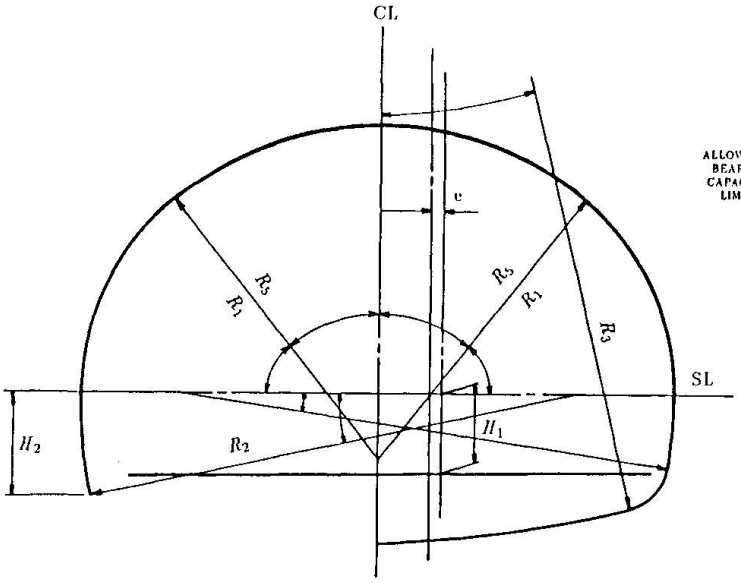
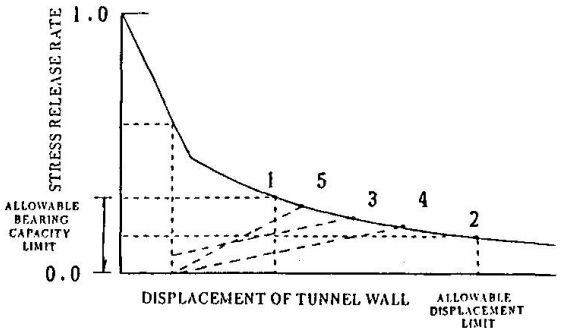


Fig. 4: Definition of dimensions of a tunnel



CALCULATION CASES

No	Calculation conditions
1	support members are subjected to their ultimate resistances
2	tunnel wall displacement is allowed up to the limit of the allowed displacement
3	rockbolt loads with a prestress equal to the pull-out capacity, shotcrete and steel ribs to behave elastically
4	rockbolts without prestress, shotcrete and steel ribs to behave elastically
5	rockbolts, shotcrete and steel ribs to behave elastically

Fig. 5: Analysed cases (Theoretical Design Methods)

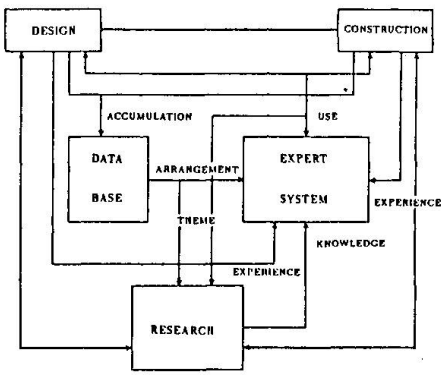
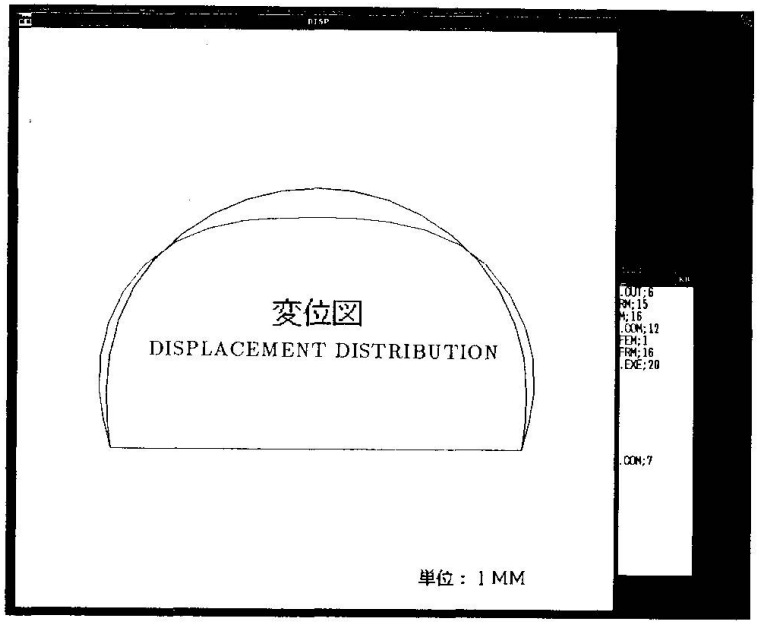


Fig. 8: Relation between the Expert System and the Design - Construction - Research Cycle



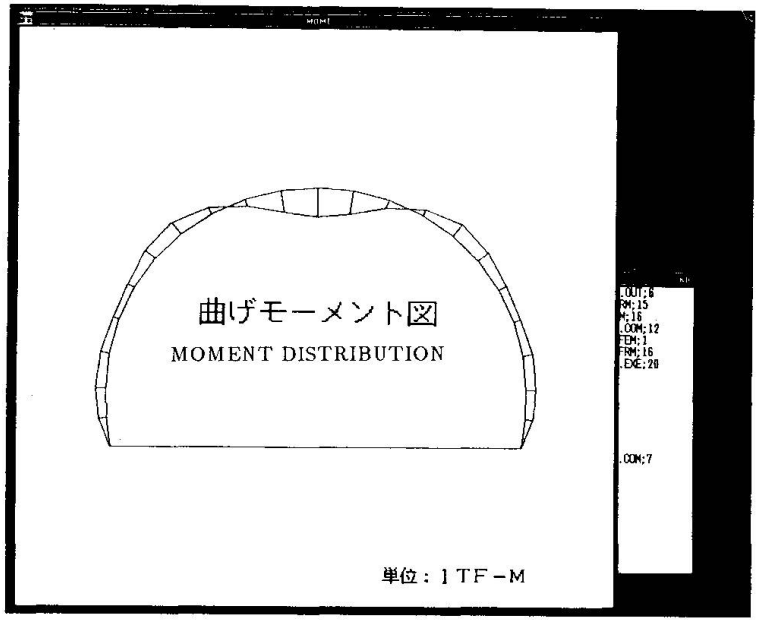
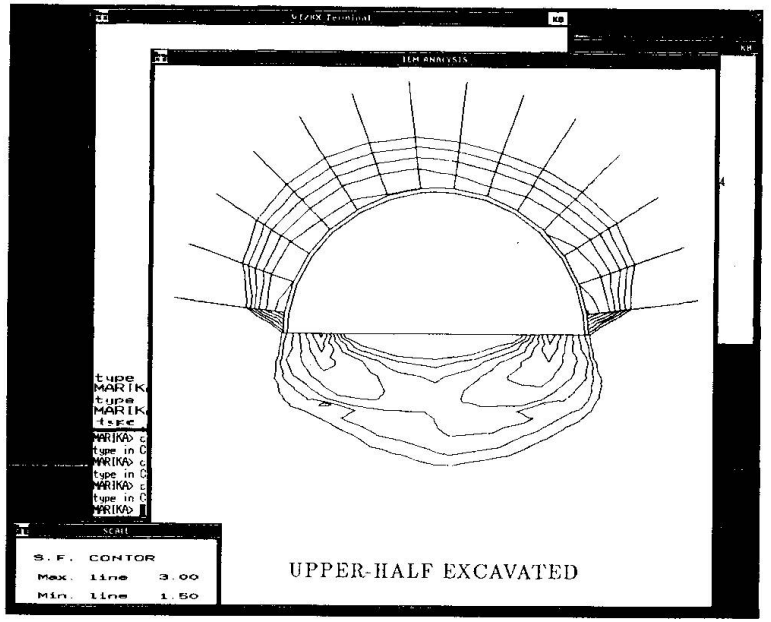


Fig. 6: Displayed calculation results (Framed Structure Design method)



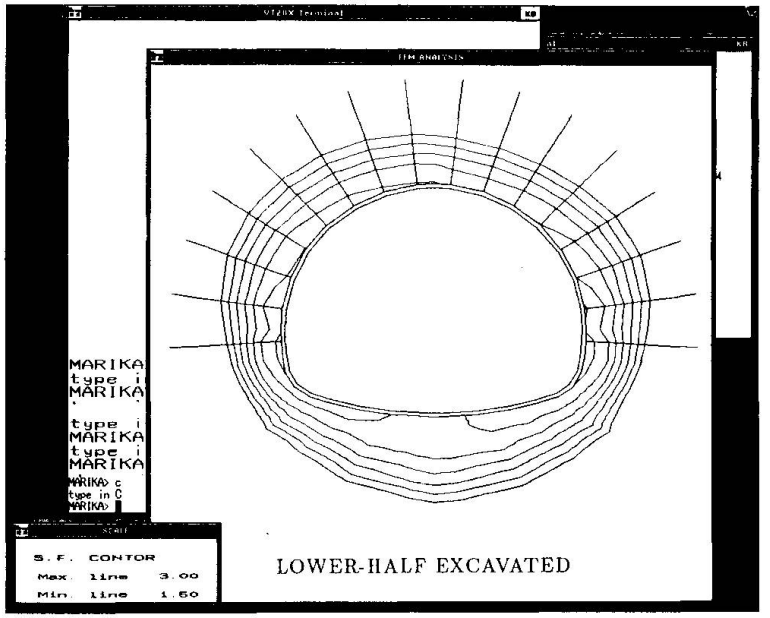


Fig. 7: Displayed calculation results (Numerical Analysis Design Method - FEM)

Valtellina Alert System: towards an Environmental Risk Diagnosis

Système d'alerte de la Valteline: analyse de risques pour l'environnement

Warnsystem im Veltlin: Diagnose von Umweltrisiken

Franco ANESA
Born in 1941, ISMES Instrumentation and System Department (DSS) manager, member of ISA (International Instrumentation Association) and AIPnD (Non-Destructive Test International Association), he is responsible for the planning of DSS activities.

Alessandro BONZI
Born in 1956, graduated in electrical engineering in 1981 at the Politecnico of Milan, he is the DSS software laboratory responsible.

Daniela LAQUINTANA
Born in 1964, qualified in informatics in 1983 at the ITIS of Bergamo, she is involved in the development of several software systems of the DSS department.

Anna PILENGA
Born in 1957, graduated in physics in 1982 at the University of Milan, she is involved in the development of several software systems of the DSS department, with special attention to A.I.

Maurizio VAVASSORI
Born in 1951, graduated in electronic engineering in 1975 at the Politecnico of Milan, he was involved in the design of special data acquisition systems. Now he leads the DSS technical projects (hardware and software).

SUMMARY
The paper describes the features of the system developed in the Valtellina area for the risk diagnosis following landslide events. In particular, it describes the development of advanced equipment that acts as intelligent assistant to the operator, supporting him in the complex activities connected with the risk diagnosis and using both graphical and numerical tools.

RESUME
Cet article décrit les caractéristiques et les potentialités d'un système réalisé dans la zone de la Valteline pour l'analyse en temps réel du risque lié aux éboulements. En particulier on décrit comment des instruments avancés ont été dévelopés pour aider l'opérateur pendant les complexes activités d'analyse et de prévision du risque en lui mettant à disposition des instruments graphiques et numériques ainsi que des modèles d'interpretation.

ZUSAMMENFASSUNG
Der Artikel untersucht und beschreibt die Eigenschaften und die Leistungsfähigkeit des Systems, das im Veltlin installiert worden ist, um die Gefahr von Rutschereignissen feststellen zu können. Die Entwicklung einer hochstehenden Instrumentierung wird beschrieben. Diese erlaubt dem Operator, die in graphischer und numerischer Darstellung anfallenden Messresultate zuverlässig auszuwerten.

1. INTRODUCTION

Following the landslides in Valtellina in July 1987, the Board of the "Regione Lombardia" and the Department of "Protezione Civile" appointed ISMES to develop systems to check and monitor the stability of the slopes effected by the landslide. On the basis of this assignment, ISMES implemented hydrogeological monitoring systems in the area of Val Pola and Valmalenco, together with the Decisional Informative System for alerting appointed officials of possible re-occurrency. Fig. 1 shows in brief the organizational-functional structure employed to acquire and interpret the monitoring data, as if was developed.

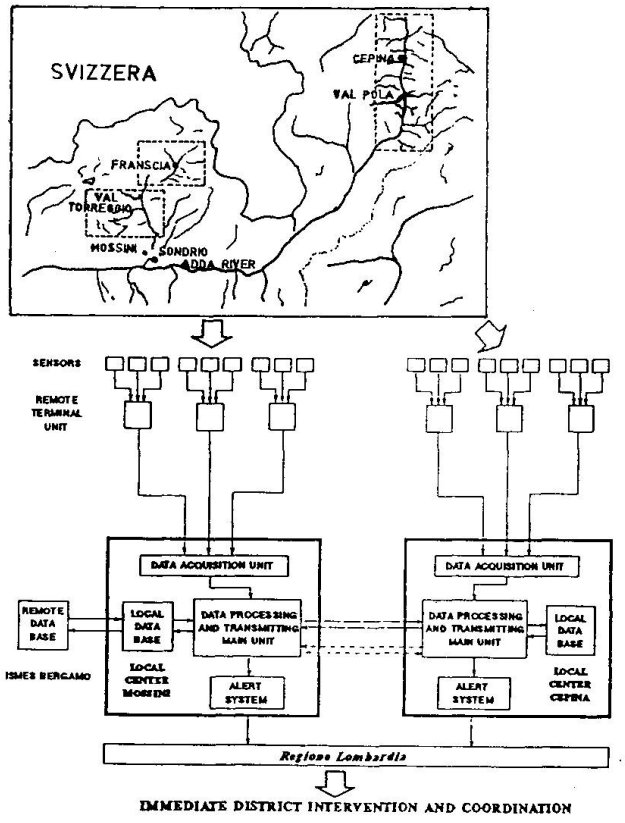


Fig. 1

The technical organizational structure includes a set of instrumental networks for controlling all the hydrogeological and climatic aspects of the site (slope instability, subsoil hydrology). The sensors are positioned at possible risk-creating area and are connected to the remote data acquisition units. The signals are trasmitted via radio to the central acquisition systems located in Cepina (for the hydrogeological monitoring of Val Pola and Presure) and in Mossini (for the hydrometeorological monitoring of Val Torreggio and Franscia, and for the hydrometeorological monitoring of Val Malenco and Alta Valtellina).

The data collected in the respective areas are processed by the Decisional Informative System for the "alert". The programs operating there check safety conditions providing a constantly updated description of the conditions of the site through the use of video terminals. The computer at the Cepina Centre is connected by telephone and radio to the analogous computer at the Mossini Centre, forming a single integrated system.

The following article describes and analyzes in detail the functioning of the Decisional Informative System for the "alert" with particular considerations to the methodology employed in its development. Such methodology characterizes the system as the first step in the evolutionary process aimed at the definition of an expert system for diagnosing the risk. The necessity of providing qualified experts, working in Valtellina, with an automatic instrument collecting and synthesizing the measurements, gave the opportunity of thoroughly investigating the theoretical study of a generalized architecture for systems supporting decisions indipendently of the specific application.

From a conceptual point of view such architecture is based on the close correlation between the moment of acquisition and the interpretation of measurements. From a practical point of view such a correlation finds expression in the employment of a model with a high level of automation.

2. WHY A MODEL WITH A HIGH LEVEL OF AUTOMATION?

The magnitude of measurements required and of geotechnical/hydrological parameters to be monitored are valid reasons in themselves to justify the use of an automatic system for the data acquisition and processing.
Additionally in order to analyze the acquired data rapidly and provide a quick diagnosis of risk situations it's necessary to have at one's disposal the use of high level tools such as territorials maps , graphs of trends and tables. These tools enable one to analyse in easy manner every information he needs.
The above mentioned requirements were verified in the development of the Decisional Informative System for the "alert" and in what has been employed within it to assure the reliability and the complete automation of the decisional process (no-break electric supply systems, remote alarm systems via radio. etc.).

3. WHY NOT A REAL "EXPERT SYSTEM"?

To answer this question it is necessary to consider the operational conditions of emergency that permitted the development of the system, but that also conditioned the methodological approach to the solution of the problem. In particular the necessity of collecting and checking the measurements on the site with the same timeliness with which the sensors were installed, together with the evolution of the situation (passing from one to three monitoring areas) hindered the possibility of projecting the system as an "expert one" carried out by means of specialized hardware and software tools. The initial approach was therefore of a traditional nature, aimed at the implementation of an automatic system for acquiring data and processing it with hardware and software tools typical of real-time precesses. During the development of the system, the conceptual problems at the basis of the "system to support decisions" took shape more clearly. Such problems have therefore changed the work environment by introducing methods and concepts close to expert systems.

4. NEVERTHELESS AN "EXPERT SYSTEM"

The complexity of the system and the variety of processes carried out tend however to define the system as an "expert" one. The procedures implemented are not based on a simple check of surpassing the threshold of the sensors, but on the automatic processing of interpretative models and therefore of predefined situations according to which the alert and mobilization thresholds are established. The system is also able to assure the surveillance and alert function by defining the whole of the probable diagnoses even in case of uncertainty or incompletness of measurements. The operator can then check such diagnoses by analyzing in detail the deductive process that the system employed automatically.
Finally, the sophisticated checks on the correctness of measurements, aimed at avoiding false or in any way altered diagnoses, emulates the behavior of the expert who critically analyzes the acquired measurements before considering them valid for the processing of alarm signals.

5. FUNCTIONAL ARCHITECTURE OF THE SYSTEM

The functional architecture of the system executed, divided into modules, is reported in the Fig.2. The complete functional block diagram is reported in Fig. 3.
The following is a detailed functional description of each single module.

5.1 Acquisition module

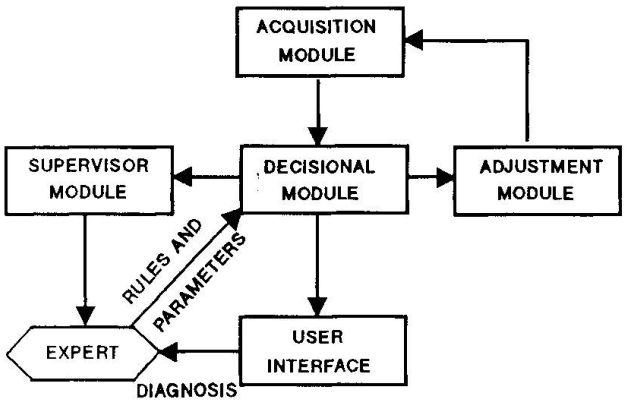


Fig. 2

The acquisition module allows the periodical and automatic collecting of measurements obtained by the different monitoring subsystems (hydrological, geotechnical). The data which are not acquired automatically are handled manually by means of controlled video masks.
The programs check the validity of measurements in order to transmit only reasonable measurements to the DECISIONAL MODULE.
The monitoring process is carried out in subsequential steps as relates to the following:

- instrumental validation, consisting of checking the correct operation of the acquisition unit, and verifying that the measurements remain within the limits both the full scale of the instruments and the physical range of the observed magnitudes.
- validation with the monitoring of tolerance bands, consisting of the verification that the acquired measurements respect the maximun allowed shifts compared to a reference function characteristic of the sensor. The reference function and consequently the tolerance bands can have a periodical or a constant trend, depending on the type of magnitude considered. For example temperature has a seasonal periodical trend, determined by a proper mathematical function; the mathematical function can be updated each time new measures join the historical data base.
- validation with the monitoring of the speed of variation, consisting of the verification that on a time base typical of every type of measurement there are no variations in the data value surpassing the pre-established thresholds. This allows the evaluating of the reliability of the measurement by filtering those values that, even though numerically correct, result in being inconsistent with the physical reality of the measured magnitude and therefore probably altered by occasional noises.

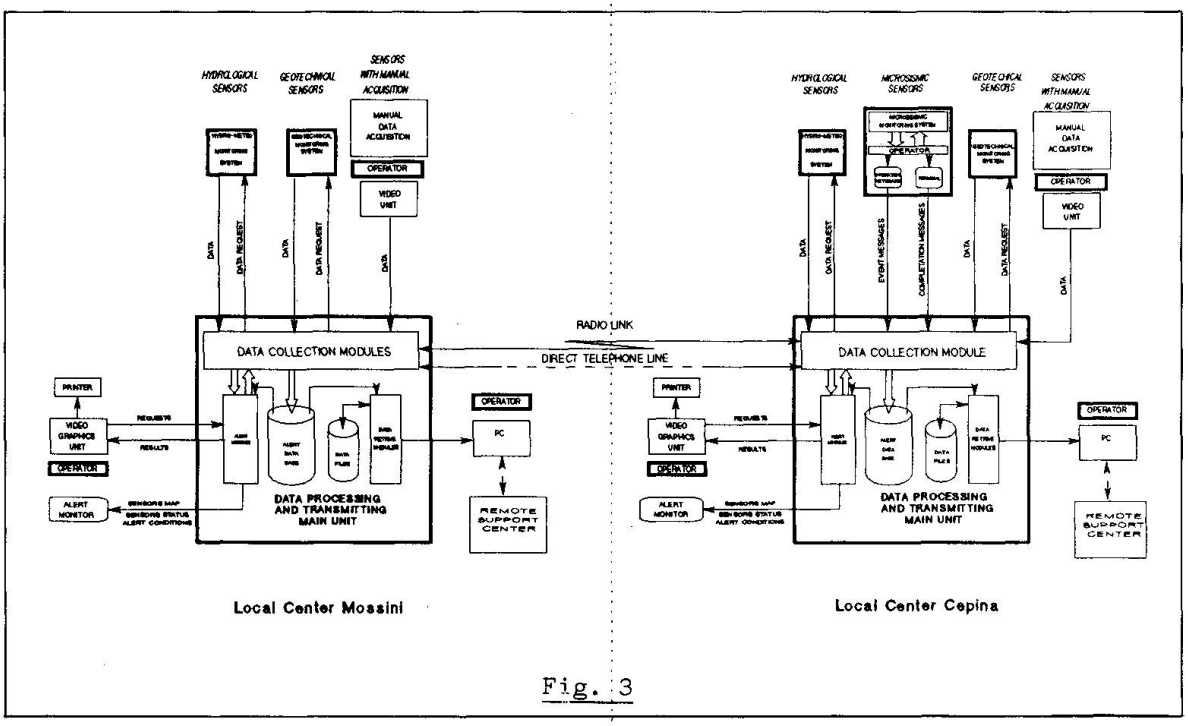


Fig. 3

5.2 Decisional and supervising module

After having been controlled and validated, the acquired data are stored in a temporary data base with a maximum storage capacity of 15-20 days.
This data base, updated in FIFO modality, is the required organizational support of the data for processing the "alert" procedures. This data base also serves in the production of graphics and reports which summarize the analyses of the measurements taken in a short period of time (15-20 days)
The storage operation consists of properly organizing the measurements received by each monitoring network, completing them with all the information required for the subsequent analyses, such as: the type of measuring network to which the data belong, the storage date, the validity codes of measurements, etc.
The interpretation of the data acquired by the sensors must emulate the capability of an expert at executing a comparative analysis of all the information received. For this reason the interpretation refers to analysis procedures, consisting of a group of rules defined by experts themselves, according to which it is possible to obtain an indication of the state of the situation.
The procedures, drawn up on the basis of possible predefined risk situations, allow the indication of dangerous conditions already present or in course. All the measures obtained by the monitoring systems are subject to controls to diagnose the possibility of surpassing the absolute measurements thresholds, thresholds of speed of variation and those of the acceleration of variation. When the system underlines the occurence of one of the risk-coded situations, it sends messages to the operator and to the monitoring systems; these are required to send data more frequently until the anomaly has stopped.
The alarms and pre-alarms are displayed on video-maps and are signalled to the operator by means of acoustic and light signals.
In the case of anomaly, the supervisor can decide on particular operations according to the information provided by the system. Such operations are parallel to the automatic procedures of adaptation that the system makes on the basis of the rules codified in the analysis procedures. This permits the possibility of manual interventions in critical or unexpected situations. When these manual interventions become repetitive, it is possible to automate them by coding the diagnosis and the consequent process.
The ADAPTATION MODULE is activated when a change in behaviour of the system itself is required.

5.3 Adaptation module

To describe how the ADAPTATION MODULE functions, the following should be considered:
- the phenomenon to be monitored often requires different frequencies of observation, depending on the phase being observed;
- the system employed for the surveillance is so complicated that it requires particular procedures in order to check its functionability.

These considerations are better explained below.

5.3.1 Different observation frequencies

In order to examine the concept deeply one can use the example of the cracking phenomena in soil and how cracks normally develop. In general, the phenomenon begins with a constant process. In the first phase, they are easy to monitor through acquisitions at pre-established intervals (usually once per hour). In subsequent phases the dynamics of the phenomenon, before the final collapse, acquire characteristics requiring acquisitions at much lower intervals (usually every 15 minutes). The problem is therefore that of being able to follow the phenomenon during its evolution automatically.

The solution could be in finding the maximum acquisition frequency in order to effectively reconstruct the evolution of any event regardless of the conditions; This could be done on the basis of an evaluation made by expert personnel or through previously established experiences.
This solution sometimes can not be implemented; in fact, when data acquisition at high frequency is not necessary, it provides the data base with an amount of useless information, slowing down the management of the whole system.
Hence one can note the importance of having integrated into the system the possibility of automatically adapting the acquisition frequency of the peripheral acquisition modules in accordance with the specific needs which arise throughout the evolution of the phenomenon. This type of operation assures the presence of only the necessary information required for a correct interpretation of the situations, at any given moment.

5.3.2 Checking the basic effectiveness of the system

Since the system is being implemented under critical conditions, such a territory subjected to landslides or other natural disasters, and since it must foresee such phenomena, it is extremely important to keep the condition of the system itself under control.
It has therefore been forecast to automatically make all the possible checks on the various hardware and software components of the system and, in case of their failure, to then be able to restore and re-adapt itself to the point of effectiveness.
With reference to Fig.4 it is possible to underline how the module "Management of Test Proceedings" verifies the correct use of the different processing software modules by operating them periodically and checking that each of them behaves as established. Each processing module is in turn responsible for the control of the peripheral units connected to it (printers, recorders, teletransmission units, etc.) through the performance of proper test sequences.
In case of any particular mal-function, the system signals and records what has been diagnose and, when possible, it attempts to automatically restore the components which are out-of-service. Examples of this are software and hardware reset, the operation of spare parts or of alternative communication channels.
In the case of non-localized damages on specific equipment, the system can pass to a partial or complete initialization of its software madules.

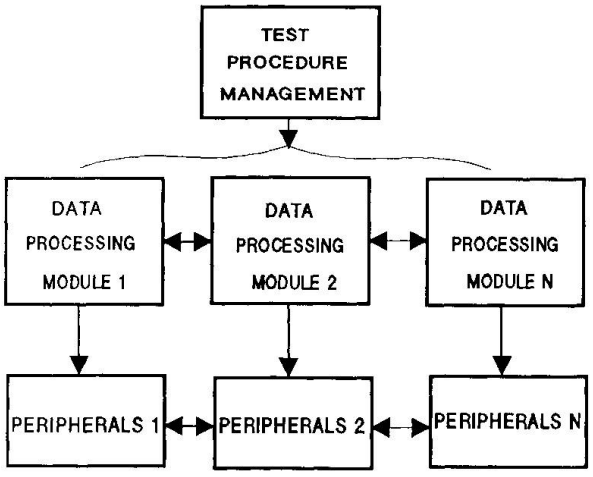


Fig. 4

6. USER INTERFACE

This consists of a set of modules through which the User can obtain all the basic information relating to the sensor measurements and to system conditions.
The data stored in the data base can be analyzed and visualized graphically or numerically on the alarm consolle. (Fig. 5)
The operator can interact with the system through selective entries protected by passwords in order to find out, visualize and, when necessary, update the informations contained in the data base.
Similarly the operator can interact with the system in order to update and

modify the data base of the channels and of their properties (qualified or non-qualified for processing of alarm procedures, manual or automatic acquisition, etc.)
The functions that the system is capable of, through the use of the map monitor video, are the following:

- viewing of topographic maps showing the areas under surveillance (Val Pola, Valmalenco) with the identification of the various zones which delineate the area and of the installed measuring positions (Fig.6, Fig.7)
- viewing, in a specific area of the monitor, of the channels in an alarm or pre-alarm state;
- viewing, in a specific area of the monitor, of the running condition of the different units integrated into the system: peripheral units, data acquisition systems, computers, etc.

Fig. 5

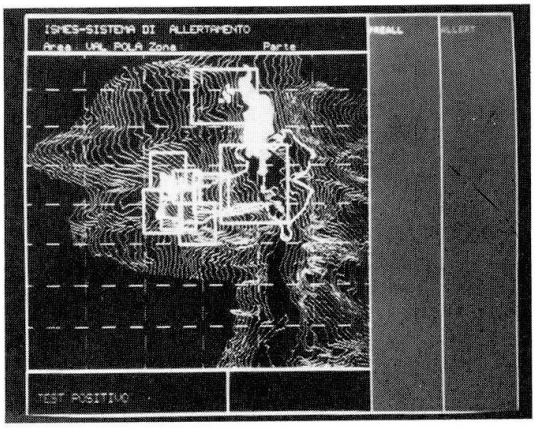


Fig. 6

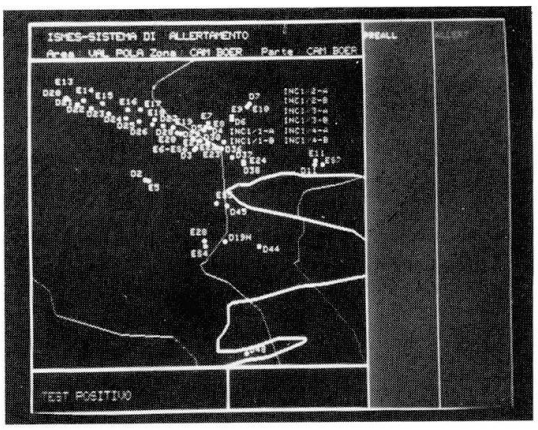


Fig. 7

7. SYSTEM EVOLUTION

It is extremely important to assure the development of all the rules that make-up the alarm procedures, through an accordance with the professional experience acquired over the years by the technicians. In order to facilitate such development, a synthetic language has been established, whose main structure is the recurrent control structure "IF-condition-THEN-action"; where the "condition" can also be a system of conditions and the "action" can either be the insertion of a further rule or the final diagnosis. The rules can be so stored by the expert by means of a program that guides him to the definition of "conditions" and "actions" according to a deductive logic, and then finally to the construction of a hierarchical structure. The program itself will control for it's consistency and completeness. "Conditions" and "actions" must be defined by a synthetic rather than a descriptive language. Hence one can note

the necessity of creating a "dictionary" of passwords covering all the possible forecast conditions and actions and a "grammar" according to which such words must be connected. The person defining the rule has the task of checking its semantic correctness; the program itself provide a series of formal checks whithin the definition, presenting satisfactory hypotheses in order to check a particular thesis, by running through the hierarchical-structure by means of "backward chaining".

Knowledge-Based Finite Element Analysis

Analyse par éléments finis par base de connaissance

Wissensbasierte Finite-Element-Berechnung

Siegfried F. STIEMER
Associate Professor
University of British Columbia
Vancouver, BC, Canada

Siegfried F. Stiemer completed his studies leading to a Dipl. Ing. Degree at the University of Stuttgart in 1972. He spent two post-doctoral years at the University of California in Berkeley before joining the Faculty of Engineering of the University of British Columbia in 1981. His various research interests range from experimental studies to computational applications in the area of steel design.

Bruce W.R. FORDE
Research Assistant
University of British Columbia
Vancouver, BC, Canada

Bruce W.R. Forde recently completed a Ph.D. degree in a cooperative programme at the University of Stuttgart, West Germany and the University of British Columbia, Canada. His present research interests focus on artificial intelligence applications to computer-aided engineering.

SUMMARY
This paper describes some innovative uses of knowledge-based techniques for problems such as process control and solution optimization. A simple displacement computation scenario is used to demonstrate the natural and flexible methods offered by knowledge-based systems for controlling interactive analysis and for dynamically formulating problem-solving approaches. Emphasis is placed on finite element analysis applications.

RESUME
Cette publication décrit quelques utilisations innovatrices des techniques de bases de connaissance pour les problèmes liés au contrôle de procédés et à l'optimisation de solutions. L'exemple simple d'un calcul de déplacement est utilisé pour démontrer la méthode naturelle et flexible qu'offre le système de base de connaissance pour contrôler les analyses interactives et pour formuler des approaches de résolution de problèmes de façon dynamique. L'accent est mis sur des applications d'analyse par élément finis.

ZUSAMMENFASSUNG
Dieser Beitrag beschreibt einige innovative Anwendungen von wissensbasierten Methoden auf Probleme wie Prozesskontrolle und Lösungsoptimierung. Ein einfaches Verschiebungsberechnungs-Szenario wird benutzt, um darzustellen wie natürlich und flexibel wissensbasierte Systeme interaktive Berechnungen und dynamische Formulierung von Problemlösungswegen ermöglichen. Der Schwerpunkt wird auf eine Anwendung der Berechnung mit finiten Elementen gelegt.

1. INTRODUCTION

Almost all existing applications of artificial intelligence in engineering belong to a group of elementary products called knowledge-based expert systems (KBES) — programs that mimic the decision-making capabilities of an expert in a particular domain. The popularity of the KBES approach can be attributed to the development of expert system shells that provide simple consultation environments with rule-based knowledge representation and manipulation. This paper demonstrates some of the more recent innovative uses of knowledge-based techniques for problems such as process control and solution optimization applied to finite element analysis.

Traditional algorithm-oriented methods constrain the logic, data structures, and procedures found in engineering analysis. Subsequent sections explain the nature of this problem and show how knowledge-based methods can simplify and improve the analysis process.

1.1. Traditional Algorithm-Oriented Methods

Finite element analysis programs usually employ algorithms. The purpose of these algorithms is to provide structured patterns that may be repeatedly applied in the solution of a specific kind of problem. This approach can provide very efficient results for rigidly structured problems that are defined in terms of simple logic and data structures such as found in many numerical processes. Early batch-job finite element software used the algorithm-oriented approach and was successful due to the rigid nature of the problem — exactly how and when to compute data is decided in advance, so the logic associated with these decisions may be rigidly embedded into the application software during the program implementation. Conversely, interactive finite element analysis programs allow the end-user to arbitrarily enter data and to experiment with alternatives, so the analysis process must be dynamically altered at run-time. Many possible combinations of data and procedures are required, so the associated logic results in an extremely large and complex algorithm.

Some basic problems with the algorithm-oriented approach are illustrated by the following simple scenario. Linear structural analysis programs typically employ a displacement computation algorithm like that shown in Fig. 1. Components of this algorithm include data ([K], {R},{D}) and procedures (*Read* Input, *Assemble* [K], *Assemble* {R}, *Compute* {D}, *Write* Output). The data may be viewed as a set of facts that determine the current problem context, and the procedures are tools that organize these facts into more useful forms.

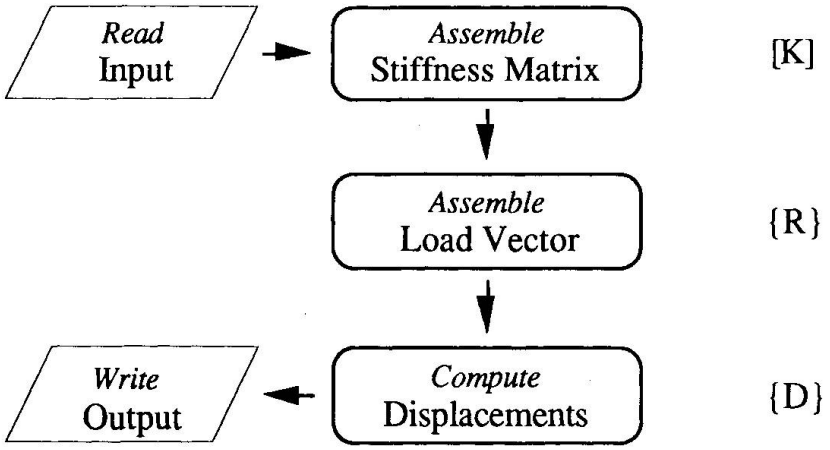


Fig. 1 Displacement Computation Algorithm.

This algorithm provides additional logic that shows how and when to invoke procedures to uncover desired facts. For example, this algorithm indicates a sequence of: *Assemble* [K], *Assemble* {R}, and *Compute* {D}. The order of activities is important and non-unique. Displacements cannot be computed before stiffness and load assembly; however, loads can be assembled before the stiffness giving another valid activity sequence of: *Assemble* [R], *Assemble* {K}, and *Compute* {D}.

Algorithm designers can arbitrarily select either of these alternatives, but in doing so they impose an unnecessary constraint on the analysis process (stiffness and load assembly are independent processes and one does not necessarily take place before the other).

In addition to the implied activity sequence, algorithms place bounds on the overall problem-solving approach. By adopting a specific algorithm, the program developer accepts the associated logic. For example, the displacement computation problem could be approached using several alternative formulations in place of the given displacement-based matrix format. Since this approach requires a stiffness matrix and a load vector, these components become part of the problem. The original logic employed by the program developer in deciding how and when computation should take place is embedded in the algorithm. If a new analysis procedure is found in the future, the developer must reexamine the logic behind the algorithm in light of the new set of alternatives.

1.2. Knowledge-Based Methods

Knowledge-based expert systems have been used in many engineering applications, but primarily as consultants or intelligent-interfaces [1, 2]. There are many other potential KBES applications within engineering software such as control and optimization. Knowledge-based approaches provide more natural and flexible methods of applying control logic for interactive analysis, as well as the potential for dynamic formulation of problem-solving approaches, that would not be possible with existing algorithms [3]. Problems associated with this new approach involve the integration of logic processing and analysis computation, as well as the representation of appropriate knowledge for the analysis domain. For example, control logic associated with the displacement computation problem can be summarized by a single rule (**IF** have [K] & have {R} **THEN** can compute {D}). A KBES that uses this rule would recognize that stiffness and load assembly are processes that take place before displacement computation (avoids unnecessary restrictions imposed by an algorithm).

Advantages of the knowledge-based approach over the traditional algorithm-oriented approach are amplified in problem areas where many options are available due to complex logic or changing technology. If the purpose of algorithms is to provide structured patterns that may be repeatedly applied in the solution of a specific kind of problem, this approach is bound to be inefficient in cases where the problem is not well defined and the desired solution pattern is constantly changing to maintain the state-of-the-art. Some aspects of the finite element analysis process remain relatively unchanged (eg. numerical solvers), yet others are constantly being improved to accommodate new problem types and solution procedures. A combination of algorithm-oriented and knowledge-based approaches is required to achieve efficient solutions in modern software.

1.3. Objectives and Organization

This paper shows that knowledge-based techniques can be used for a variety of tasks in the finite element analysis process aside from the popular roles as consultant and interpreter. Some of the problems examined include:

(1) Potential KBES applications to finite element analysis.
(2) Knowledge-based expert systems for process control.
(3) Dynamic formulation of problem-solving approaches.

The second chapter of this paper identifies a variety of potential KBES applications to finite element analysis including: consultation, interface, control, and optimization. These topics involve logic and decision-making that is easily handled using heuristics in place of traditional algorithms. Simple descriptions are given for each application in terms of the problems handled and the potential computer software configurations.

The third chapter of this paper examines a simple KBES that controls an interactive finite element analysis program. The kinds of data and procedures found in the analysis process, the relationships between these components, and the factors that make the knowledge-based approach attractive are briefly introduced. Some basic methods are discussed for rule-based representation, followed by a summary of alternative techniques for reasoning and a comparison of conventional expert systems to a KBES used to manage an interactive computation process.

The fourth chapter of this paper discusses the potential implementation of a KBES for dynamic formulation of problem-solving approaches. Rather than relying on any single algorithm with its predetermined set of procedures for computation, the proposed system employs logic from applicable technologies to construct its own algorithms, data structures, and procedures as required to achieve the best solution for any given problem.

2. KBES APPLICATIONS TO FINITE ELEMENT ANALYSIS

This chapter identifies a variety of potential KBES applications to finite element analysis including: consultation, interface, control, and optimization. These topics involve logic and decision-making that is easily handled using heuristics in place of traditional algorithms. Simple descriptions are given for each application and the potential computer software configurations.

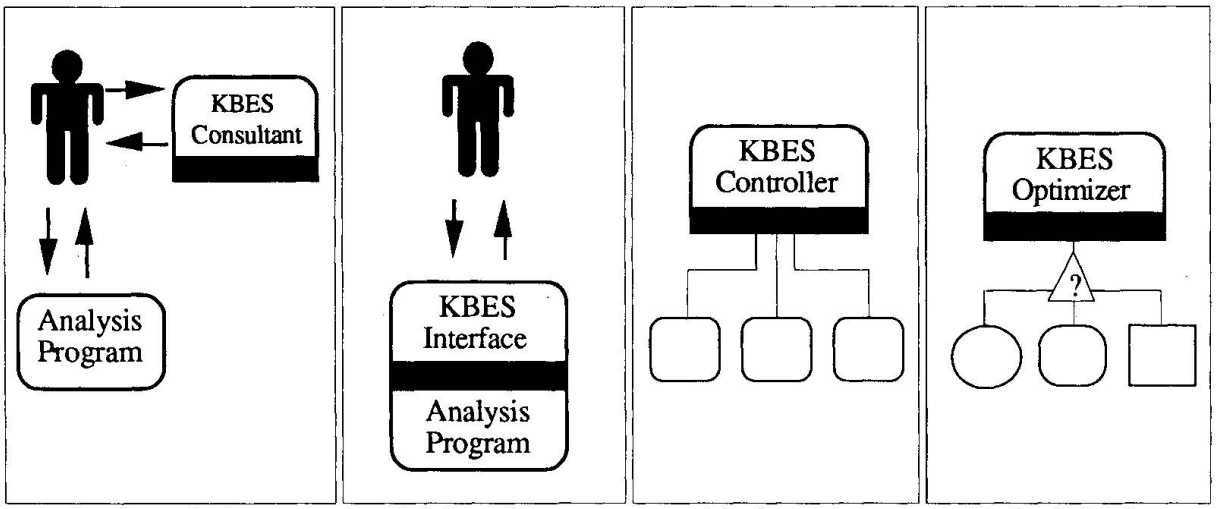


Fig. 2 KBES Applications to Finite Element Analysis.

2.1. Consultation

One of the first applications of knowledge-based technology in engineering was the introduction of automated consultants that help engineers in the use of complex computer programs. Engineering software training often involves on-the-job expert consultation to supplement the explanations given in computer manuals. Experienced engineers often do not have time to answer questions for junior personnel, so an automated consultant as shown in Fig. 2(a) is desired to work in parallel with the user of an analysis program. Early systems such as SACON [4], functioned in this way to help explain input and output files for traditional batch-oriented finite element analysis. Expert system shell consultation environments, with facilities for representation and manipulation of rule-based knowledge, are the most popular KBES technology used in engineering.

2.2. Interface

A second generation of knowledge-based technology emerged in response to problems with the consultation approach. These systems recognized that automated consultation is not always effective due to limited understanding of complex conventional application software. Even with excellent advice from a consultant, engineers inevitably find difficulties with program operation. A KBES can provide a simplified interface to engineering software [5]. Systems such as shown in Fig. 2(b) translate user-requests into facts understood by the intelligent interface. These facts are used in conjunction with a knowledge base and an inference mechanism to logically evaluate and implement operations that will achieve user-requests.

2.3. Control

Knowledge-based technology has recently been applied to problems that are traditionally handled by algorithms within conventional software. One example application is as the controller of an interactive finite element analysis program [6]. The system shown in Fig. 2(c) uses a KBES to select and apply a variety of analysis computations. Interactive software was made popular by the spreadsheet concept — changes made in one cell automatically propagate throughout the spreadsheet. Similar capabilities in an analysis program require integration of modelling, solution, and interpretation activities. Computation of all results for each change is not necessary and is impractical in terms of real-time operation, so some decisions must be made at run-time. The KBES approach is attractive in this case because only a few simple heuristics are required to describe the analysis process even for an arbitrary number and order of actions. Conversely, equivalent algorithm-oriented approaches would be more complex to implement and even more difficult to improve or upgrade in light of new facts or strategies.

2.4. Optimization

Engineering technology relies heavily on experience to assure that the best solutions are used to obtain the best end-product. Heuristics or rules-of-thumb are traditionally incorporated within engineering software to achieve practical results, yet the solution procedure itself is defined by a specific fixed algorithm. Optimization of the end-product is often equated with the selection of a few possible alternatives followed by evaluation and comparison using objective functions. This approach can be built into algorithms, but doing so constrains the potential solution domain. KBES technology makes it possible to perform optimization without limiting the number or type of alternatives, as the data defining facts and strategies may be part of a changing knowledge-base. Although product design optimization can be improved using KBES approaches, this paper focuses on solution optimization that was not possible using algorithm-oriented methods. Unlike traditional programs that simply select and apply a variety of analysis procedures, the system shown in Fig. 2(d) uses a KBES to dynamically formulate the problem-solving strategy. A finite element analysis program operating with this paradigm can decide when and how to store and compute data. Whereas traditional software employs a "brute-force" approach for many numerical problems, new systems can use knowledge to take advantage of common computations and to share results as humans would do when solving problems by hand calculation.

3. KNOWLEDGE-BASED PROCESS CONTROL

Control logic can be separated from analysis software, and a knowledge-based expert system can use this logic to perform interactive computation. Traditional algorithms for finite element analysis can be replaced by a few simple heuristics that define how and when computation should take place. Rule-based techniques are used to represent and to reason about this analysis domain knowledge. Subsequent sections explain why the KBES approach is more effective and flexible than traditional algorithm-oriented approaches.

3.1. Domain Knowledge

Finite element analysis is a three-stage process of: modelling, solution, and interpretation [3]. The components of a simple analysis problem vary depending on the particular domain, so this paper will restrict discussion to the example shown in Fig. 3.

Knowledge related to modelling and interpretation deals with questions such as how to select elements that will properly represent the physical problem, and how to recognize when numerical results correspond to real behaviour. These questions are subjective and difficult for junior engineers to answer, as they rely heavily on experience, so early KBES consultants focused on heuristics of this nature. On the other hand, control is a relatively simple topic. In order to achieve a solution, analysis data must be used in structured processes. As in mathematics, explicit logical answers are often available for questions about control. For example, stiffness and loads must be assembled before displacements can be computed. Such knowledge can be easily represented using rules and then used by an inference mechanism to reason about specific problems.

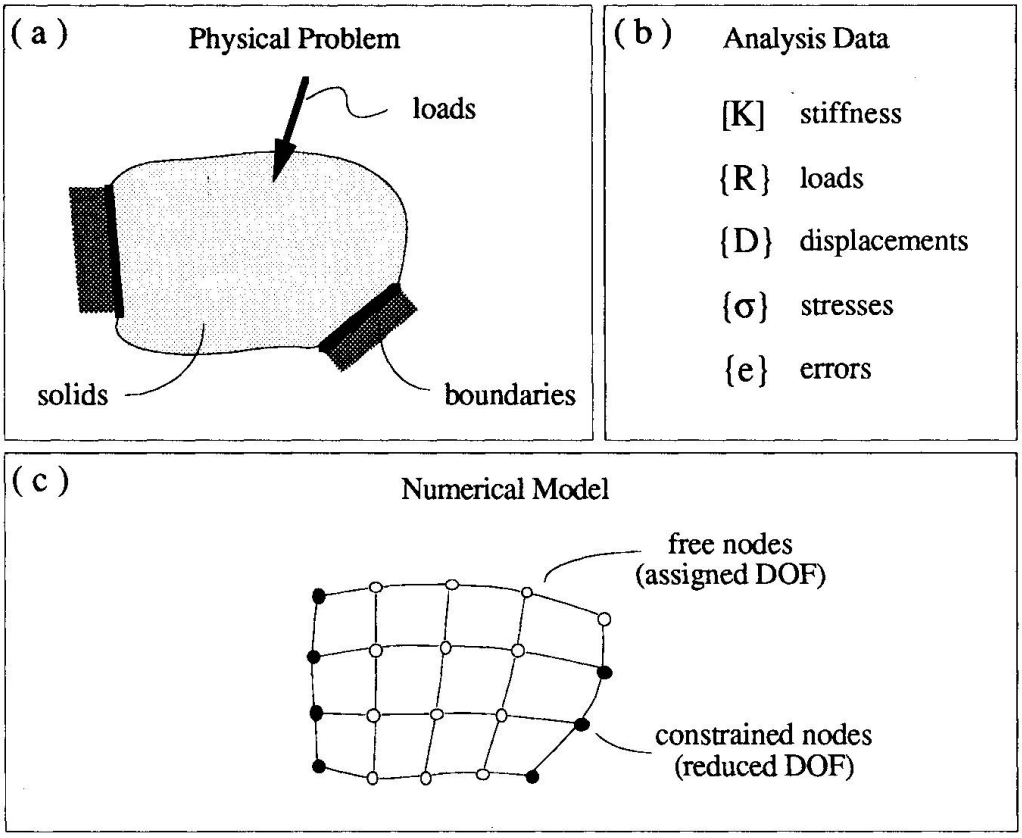


Fig. 3 Components of a Simple Analysis Problem.

3.2. Representation

Engineers that are familiar with conventional approaches recognize that they have encoded their knowledge into the algorithms that control their programs. There are two major problems with this approach. First, expert knowledge (facts or rules) that is built into an algorithm cannot be questioned or updated by the program. Second, traditional data structures and procedures used to describe the problem or its solution place unnecessary constraints on the problem-solving approach.

Systems are called "knowledge-based" if they separate domain knowledge from the program control structure. This allows KBES programs to question and to update their own knowledge base of facts and rules. Most engineering expert systems do not actually employ machine learning with automatic updating of knowledge bases, but explicit definition of domain knowledge is a step in this direction in comparison to previous approaches.

Knowledge about the problem-solving approach is often subtly implied by the data structures and procedures linked to algorithms. For example, algorithms found in finite element analysis often make references to matrices that hold information about the problem. Although these data structures may be arbitrarily sized, the fact that there is any mention of matrices implies the kind of solution. Object-oriented programming is becoming popular because it encapsulates data and procedures so that problem-specific information may be placed in packages. A similar trend is appearing in the design of knowledge-based systems leading to islands of knowledge with local KBES management.

3.3. Reasoning

Most expert systems use a form of deductive inference to reveal facts. Some theorem-proving systems start by examining rule premises. If all conditions in a rule premise have been satisfied, the rule hypothesis may be deduced. This new hypothesis may be used with other known facts to prove additional rules. Since this approach works directly through the rules from conditions to hypotheses, it is known as "forward-chaining" inference. An alternative strategy involves searching rule hypotheses for a desired goal. Once a rule has been located with the appropriate hypothesis,

each of its conditions are examined. Conditions that are not proven are added to the list of goals. If all conditions have been proven true, the rule hypothesis is also proven true. Eventually, the original goal is achieved or additional information is requested. Since this approach works from hypotheses back to conditions, it is known as "backward-chaining" inference. Real expert systems employ a mixture of the two approaches.

Reasoning is performed by an inference engine as shown in the knowledge-based control portion of Fig. 4. Replacing the logic that was embedded into a traditional algorithm-based approach with explicit facts and rules in a knowledge-based system allows program developers to construct applications around a generic event-driven core (a popular paradigm for interactive software). Instead of invoking computation in a rigid order, as mentioned earlier, the KBES approach can make logical decisions based on the current problem context and knowledge base.

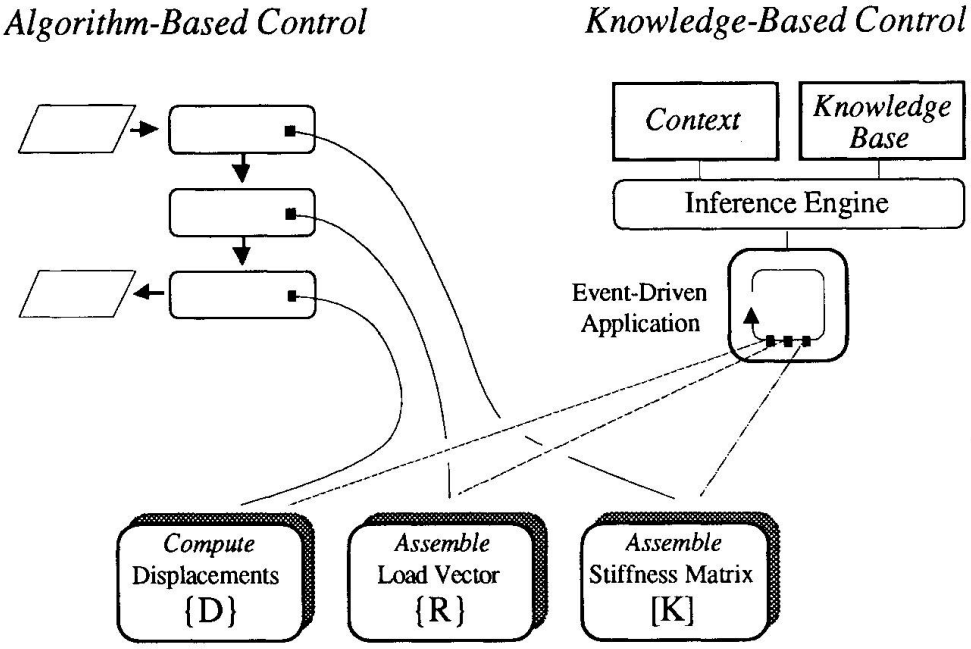


Fig. 4 Alternative Control Paradigms.

4. KNOWLEDGE-BASED SOLUTION OPTIMIZATION

Analysis programs designed and implemented by experts may provide efficient solutions to current problems; however, increasing specialization and advancing technology will eventually make it impossible to achieve optimal analysis using traditional methods. The problem is that algorithm-oriented techniques require software developers to formalize the complete analysis process in terms of a specific problem-solving strategy, a limited number of procedures, and a predetermined set of data structures. The proposed solution is to automate the selection and application of low-level analysis components using a knowledge-based system that can organize and solve equations in the same manner as an expert.

For a simple finite element analysis program, this involves the representation of equations for stiffness, displacement, stress, and error computation. Using its knowledge base, the equation-solving expert-system should be able to identify: the order in which the equations are applied, where the data is stored, and what calculations can be used in multiple contexts (eg. stiffness, stress, and error computations share certain intermediate matrices). This new approach must take advantage of the significant number of efficient general-purpose procedures currently available for conventional software development. Object-oriented programming can be used to create a hybrid environment that integrates the best features of KBES technology for reasoning and representation with the existing base of numerically efficient procedure-oriented software.

4.1. Object-Oriented Design

Computer software development involves the implementation of an application or concept in a programming environment. Finite element analysis programs are usually built around concepts of nodes, elements, stiffness matrices, etc. Although software of this kind is often called modular and reusable by its developers, this is usually not the case. Conventional finite element procedures (subroutines) assume many facts about their data structures. A typical assumption may be that nodes have a certain number of degrees of freedom. In the event that future software has a different kind of node, many procedures may not be used without modification. The intention of object-oriented programming is to encapsulate data and procedures into useful packages that provide a higher degree of modularity than traditional software. These packages are the building blocks used to dynamically construct solutions based on the problem context [3].

4.2. Optimization Strategy

Rather than starting program construction with a fixed problem-solving strategy in mind, a knowledge-based system can be used to evaluate the suitability of available object-oriented components and to assemble them into a specialized form for any given problem. Conventional programs place emphasis on designing a robust algorithm that will efficiently solve a specific problem type. Conversely, this system places emphasis on general representation and reasoning capabilities. Domain knowledge, including the problem-solving strategies, is maintained separately from the expert system shell so it can be modified during the solution process.

5. CONCLUSIONS

Knowledge-based techniques can be used to improve a variety of tasks in the finite element analysis process. Although traditional algorithm-oriented approaches are useful for rigidly structured problems, modern interactive computer software requires more flexible and efficient methods for handling data, procedures, and associated logic.

Simple rule-based systems can be used in innovative applications such as process control and solution optimization. KBES programs are well-suited for these problems because they explicitly separate facts and rules from the program control structure, allowing both the analysis results and the procedures used to achieve them to be scrutinized.

REFERENCES

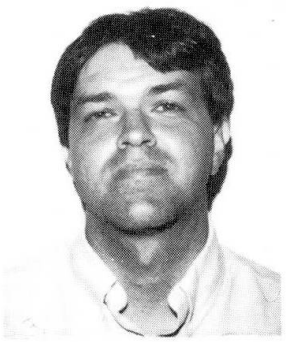
1. Adeli, H., Ed. *Expert Systems in Construction and Structural Engineering*. New York, NY, USA: Chapman and Hall, 1988.
2. Allen, R.H. *Expert Systems in Structural Engineering: Works in Progress*. Computing in Civil Engineering **1**, 312-319, 1987.
3. Forde, B.W.R. *An Application of Selected Artificial Intelligence Techniques to Engineering Analysis*. Ph.D. Thesis. Department of Civil Engineering, University of British Columbia, Vancouver, B.C., Canada, 1989.
4. Bennett, J.; Creary, L.; Engelmore, R.; Melosh, R. *SACON: A Knowledge-Based Consultant for Structural Analysis*. Technical Report STAN-CS-78-699, California, USA: Stanford University Press, 1978.
5. Weiss, S.; Kulikowski, C.; Apté C.; Uschold, M.; Patchett, J.; Brigham, R.; Spitzer, B. *Building Expert Systems for Controlling Complex Programs*. Proceedings of the Second National Conference on Artificial Intelligence (AAAI-82), Pittsburgh, Pennsylvania, USA, 322-326, August 1982.
6. Forde, B.W.R.; Stiemer, S.F. *Knowledge-Based Control for Finite Element Analysis. Submitted to:* Engineering with Computers, 1989.

Alternative Programming Techniques for Finite Element Program Development

Autres techniques de programmation pour le développement de programmes d'éléments finis

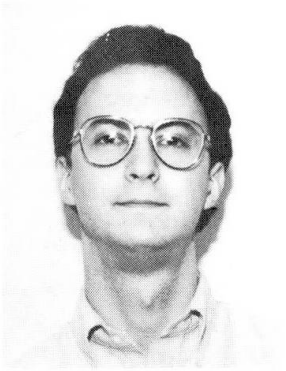
Alternative Programmiertechniken für die Entwicklung von Finite Elemente Programmen

Daniel R. REHAK
Assoc. Professor
Carnegie Mellon University
Pittsburgh, PA, USA



Daniel R. Rehak, born in 1951, received Bachelor's and Master's degrees in Civil Engineering from Carnegie Mellon University, and a Ph.D. from the University of Illinois. His research interests center on the applications of emerging computer technologies to large-scale engineering software development, especially in the civil engineering domain.

John W. BAUGH Jr.
Graduate Res. Assist.
Carnegie Mellon University
Pittsburgh, PA, USA



John W. Baugh Jr., born in 1960, received the Bachelor's degree from Auburn University, and the Master's degree from Carnegie Mellon University, both in Civil Engineering. He worked as a Research Engineer in structural mechanics at Battelle, Pacific Northwest Laboratory before returning to pursue the Ph.D.

SUMMARY

Finite element program development is hard; the translation of a page of matrix algebra, integrals and derivatives into code results in several tens of thousands of lines of non-trivial code. The difficulty arises because implementations specifiy *how* to solve the problem, rather than *what* the solution entails. Alternative programming approaches, based on formal specifications and data abstractions, let the programmer deal with a more declarative and abstract representation of the finite element solution process. These techniques are being used in an objet-oriented environment to provide a tool-kit for researchers implementing new finite element programs.

RESUME

Le développement d'un programme d'éléments finis est complexe; la traduction d'une page d'algèbre matricielle, d'intégrales et de dérivées en codes résulte en plusieurs dizaines de milliers de lignes de codes non-triviaux. La difficulté survient du fait que des réalisations spécifient *comment* résoudre le problème, plutôt que *qu-est-ce que* les solutions impliquent. D'autres approches de programmation, basées sur des spécifications formelles et sur des abstractions de données, laissent le programmeur travailler avec une représentation plus spécifique et abstraite de la méthode de résolution par éléments finis. Ces techniques sont couramment utilisées dans un environnement «orienté objet» constituant ainsi une «trousse à outil» pour les chercheurs mettant en oeuvre de nouveaux programmes d'éléments finis.

ZUSAMMENFASSUNG

Die Entwicklung von Finite Elemente Programmen ist schwierig. Die Übersetzung einer Seite von Matrix Algebra, Integralen und Ableitungen in Programm iersprache resultiert gewöhnlich in zehntausenden von nicht trivialen Zeilen. Die Schwierigkeiten entstehen, weil die Implementierungen spezifizieren *wie* das Problem zu lösen ist, und nicht *was* die Lösung zur Folge hat. Alternative Programmierwege, die auf formalen Spezifikationen und Datenabstraktionen basieren, erlauben es dem Programmierer mehr mit deklarativen und abstrakten Darstellungen der Finite Elemente Lösung zu arbeiten. Diese Techniken werden in einer objektorientierten Umgebung verwendet, um dem Forscher ein Werkzeug für die Entwicklung von neuen Finite Elemente Programmen zur Verfügung zu stellen.

1 Introduction

The finite element method is an extremely powerful and popular analysis tool which is used in a variety of engineering domains, including mechanical and civil structures, aerospace, electrical fields, nuclear, and shipbuilding. The basic description of the method, in terms of general formulations, element derivations, material models, solvers, etc., can be presented in terms of simple and elegant mathematics. These concise descriptions (often on the order of one page of mathematics) belie the computational complexity of the method. Programs which implement the method are complex, both in their structure and in their computational requirements. There is a dichotomy between the elegance of the basics of the method and the "dirty code" used to realize programs.

Development of finite element programs, simply put, is *hard*. Programmers have to deal with a variety of issues. First they must translate the mathematics of the method into numerical procedures and algorithms, coupled with appropriate data representations. In doing this, they must handle problem domain issues, e.g., selection of material models, shape approximations, order of integration, etc. The problems of the method and the domain are only a part of the complexity. Programs must execute on real machines (often with special architectures) with finite resource limits. Complex data structures and memory and storage management obscure other parts of program. Lastly, from the user's perspective, the goal is problem solving. Inputting a problem description and reviewing results are most important, and a major portion of any program must deal with user interfaces.

These characteristics result in large, complex programs, often measured in tens-of-thousands of lines of (convoluted, unmaintainable, nonportable) code. This situation is mitigated by the programming methodology used. Current programs are imperative—"word at a time". All details of the method, domain characteristics and resource management issues have to be stated in explicit detail (specifying how to move every word of problem data through the solution process, one simple operation at a time). This is quite different from the general, *abstract*, mathematical statements used to represent the method. The mathematics present *what* the finite element method is, while a program describes *how* to use the method in problem solving.

The central thesis of this work is that this distinction of *what* from *how* is the cause of much difficulty in implementing finite element programs, and that alternative (non-traditional) programming techniques can be used to lessen this distinction. In terms of finite element programs, the goals are:

- to make programs more *declarative*, such that they represent more of the abstract statement of the finite element method and less of the details of how to process the data;
- to lessen program development effort by letting programmers work at a higher, more abstract level, closer to the description of the method, again without dealing with low-level implementation details;
- to improve program reliability by placing responsibility for determining many of the details of the implementation with the program (or programming environment) instead of with the programmer; and secondarily
- to exploit parallelism (prevalent in emerging hardware systems) by uncoupling flow control from implementation.

The programming techniques used to reach these goals, along with an overview of the work to date, are presented in the sequel. The approach to developing abstract, declarative programs arises from, and is influenced by, the general methodology of knowledge-based systems and artificial intelligence. In both cases the goals are the same—to provide a declarative representation of the components of the problem-solving domain and to let the computer use those representations as needed. The actual techniques used in the work come from the domains of artificial intelligence, programming languages and software engineering.

2 Techniques for Finite Element Programming

There are a variety of alternative programming techniques available which can be of value in developing finite element programs. Before giving an overview of those being considered, it is necessary to outline the requirements placed on the technologies. As stated above, the overall objective is to develop a programming approach which concentrates on telling the computer what to do, and letting it decide how to do it. To meet this objective, the technologies used must fulfill three requirements. They must provide:

- representation—what type of knowledge of the finite element method can be expressed;
- expression—how the knowledge is expressed; and
- use—how the information is used to solve a problem.

A brief overview of each of the key techniques being explored follows, including a discussion of their role in finite element program development. While not explicitly described, the three requirements described above are considered in determining the applicability of the techniques.

2.1 Abstraction

Programming methodologies have evolved from simple (unstructured) programming through structured programming (which emphasizes the decomposition of programs into procedures and data structures) to data abstraction. Simply put, abstraction is the hiding of (appropriate) details. In terms of programming, abstraction consists of developing a set of (*abstract*) *data types* and the set of *all* operators associated with the data type.

Thus an abstraction consists of a data item or entity (akin to a data structure) and the procedures which operate on the data. The key is the inseparability of the entity from the operators. What is hidden is *all* internal details of how the data is actually represented and how the operators perform their tasks. The abstraction presents an external view of what it represents (i.e., the data type) and what it computes (i.e., the operators).

An abstraction is defined by a specification of what it represents and what it does. A program is built by using operators of the abstractions to perform tasks and manipulate the data entities represented by the abstract types. The actual implementation of the abstraction (representation and operators) is totally hidden, and can be changed without impacting other parts of the program as long as the implementation conforms to the specification.

As a simple example, consider an abstraction which represents a strain-displacement matrix for an element. Some operator might return the matrix. Internally the operator might compute the entire matrix each time it is needed, or compute and save it the first time it is requested and return the stored version when needed. Similarly, for a 2–D plane-stress problem, the abstraction might represent the matrix as a two-dimensional array, or as the two unique terms. What matters is that on demand, the abstraction will return the matrix: how it decides to perform the task or represent the data is immaterial to the user of the abstract type.

Abstraction techniques directly address the major stated objective of separation of *what* from *how*. The implementation provides, but hides, the details of how to do things, while the specification of the abstraction provides a (formal) definition of what the abstract type represents and what operators do. Abstractions are still lacking in that the specification is not (necessarily) executable, but must be transformed into running code. Thus a specification does not provide all of the expressional power needed to develop a fully declarative programming environment.

2.2 Dataflow Representations

In terms of this work, dataflow is a representational strategy (not to be confused with dataflow or reduction hardware architectures, although the associated problems are the same). Dataflow provides a graphical representation of problem data dependencies (i.e., what data items are produced and consumed by each computational step). Dataflow represents only the essential temporal constraints on data computation.

A dataflow graph does not represent a single sequence of *control flows* through a set of computations. Rather, some process must interpert the data dependencies in some fashion to drive computations. Thus a dataflow representation provides a declarative form for representing how to sequence the computations in a finite element program.

Dataflow representations can be processed either in a forward (data-driven) or a backward (demand-driven) manner. In forward or *dataflow* processing, computations proceed when data is available. Results propagate, (i.e., data *flows*) through the graph. As soon as an operator has all of its ingredient data, a computation is performed and the result is propagated to downstream operators.

In backward or demand processing, computations are invoked only on demand. When a data item is needed, the operator which produces it is invoked. If the ingredient data is available, the result is computed. If it is not available, a recursive process is used to compute that item. Demand-driven and data-driven dataflow are analogous to backward and forward chaining in rule-based systems.

Besides providing a declarative presentation of control, dataflow representations can be used to implicitly represent the inherit parallelism in finite element computations. Consider the control problem of stiffness matrix assembly and solution. A few of the alternative control strategies include:

- Sequentially form all element matrices, sequentially assemble the generated matrices, then solve the entire set of equations;
- Sequentially form and assemble matrices one element at a time, solve after all matrices have been assembled;
- Sequentially form and assemble matrices one element at a time, solve after any row is complete;
- Form element matrices in parallel, sequentially assemble the generated matrices, then solve the entire set of equations; or
- Form element matrices in parallel, synchronize, assemble the generated matrices in parallel, synchronize, then solve the entire set of equations (serially or in parallel);

There are other alternatives, including any of the above done on a nodal or degree of freedom (DOF) basis instead of an element basis.

A dataflow graph which represents the assembly and solution process for static analysis is shown in Figure 1 (individual nodes of this graph can be further decomposed). The processor which interprets this graph can implement any of the control strategies described above. Based on the data types used, the same graph represents the solution at various levels of data granularity (e.g., element, node, or dof). Thus dataflow representations provide a powerful declarative technique for uncoupling control flow from program structure and hardware architecture.

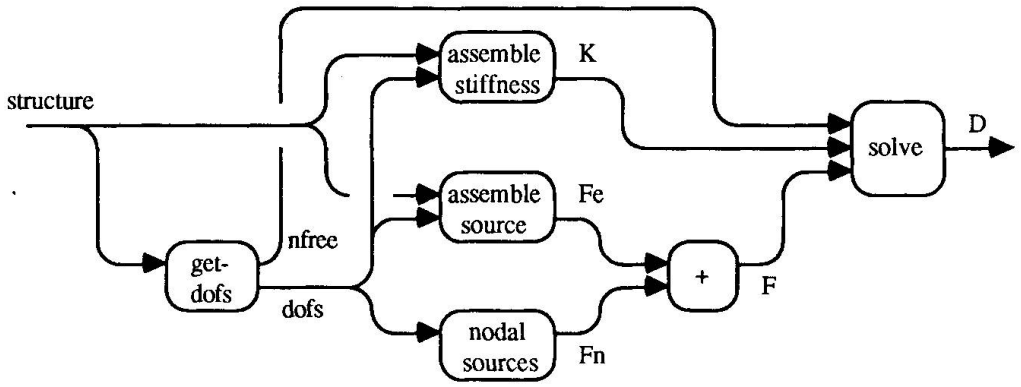


Figure 1: Static Analysis Dataflow Graph

2.3 Functional Programming

Functional programming is another paradigm which might be used to represent a finite element program. In functional programming, all expressions are of the form of a function applied to data items: functions are the primary entities in the language. The structure of a functional language is based on λ-calculus (*pure* Lisp is an example). The languages use abstraction to represent the data types, and binding in combination with function invocation to perform computations.

Functional programming provides a declarative, *applicative* form of representing problem solving. This form of a program excludes assignments, side effects, loops, branching, and representation of *state* (i.e., the program cannot rely on any knowledge of computational order). Used directly, functional programming can be used to provide another declarative representation of the finite element process.

In addition, the order of execution need not be that of simple sequential evaluation. Rather, the functional representation can be considered as one which encodes parallelism. A functional program can be compiled directly into a dataflow representation. Thus the functional form can be processed to yield either a demand-driven or data-driven problem-solving order.

2.4 Constraint Programming

Constraint programming provides yet another declarative representation strategy. Constraints can be used to represent desired relationships between data objects. As such, a constraint need not be satisfied, but if not satisfied, the fact that there is a violation is known. In addition, a constraint may, or may not, represent the needed information used to insure its satisfaction.

For example, a constraint might state that two elements must be in the same coordinate system for their stiffness matrices to be added. Such a constraint is sufficient to check that the condition exist before the elements are added, but does not provide any information on what to do if the condition is not true (e.g., apply a rotational transformation to one of the matrices).

The most desirable representation of constraints is a *nondirectional* one. For example, the relationship *area* = *width* × *height* is not an assignment statement to compute the area, but rather a relationship which can be used to compute either *area*, *width*, or *height* given the other two values.

Constraints can be used either locally or globally. In a local representation, individual constraints are used to verify pre-conditions (and post-conditions) of individual operations. In a global representation, a set of constraints can be used to represent all of the relationships which must hold in solving the problem. In this case, the entire problem can be solved by applying some constraint satisfaction procedure (e.g., *relaxation*), or the constraints can be used in a forward inference (e.g., greedy) strategy or a backward (e.g., lazy) search to drive computations as in dataflow representations.

2.5 Knowledge-Based Methods

The last declarative representational technology considered is the general area of knowledge-based methods. Of particular interest are knowledge representation strategies, such as rule-based programming. Rules provide an explicit declarative representation of problem-solving knowledge. Frame-based representations provide similar capabilities.

Different types of knowledge might be encoded in a knowledge representation. *Causal* knowledge represents the basics of the finite element method. Constraints (as described above) represent relationships and can be encoded in a knowledge representation. *Process* knowledge represents details of problem solving and constraint satisfaction (e.g., a rotation transform is used to align coordinate systems).

An important use of knowledge is as (rule-based) meta-knowledge used to control the details of the problem-solving process. For example, meta-knowledge might be used to select one type of algorithm or representation from those available (e.g., a non-sparse matrix representation is appropriate for a "small" problem). In addition to the rules used to make such selections, other knowledge must be represented, such as the characteristics of the available solution mechanisms or data representations, or the resource utilization characteristics of processes (used in making task-to-processor assignments in a multi-processor environment).

2.6 Object-Oriented Programming

As a technique for finite element program development, object-oriented programming is the implementation methodology. It provides the expression of the operational program. Object-oriented programming is a desirable implementation methodology due to four characteristics it provides: (1) data abstraction, (2) data type completeness, (3) inheritance, and (4) polymorphism.

One of the most fundamental characteristics of an object-oriented language is that the language forms and programming style provide and encourage abstraction. An *object* consists of a (hidden) local representation and an associated set of procedures which manipulate the object to produce result objects used by other operators. As such, an object is an instance of an abstract data type.

Data type completeness is an important characteristic which is missing from some "object-oriented" programming languages (e.g., C++). Data type completeness results in a situation where every entity manipulated by the program is treated equally, and can be used in any situation (e.g., can be assigned to, passed as a parameter, returned from a function, or used as components of other data structures). Object-oriented systems which provide this capability often do so by providing only a single underlying concept, the *object*. (Data types, operators, the compiler, data representations, etc., are all objects.) This approach is beneficial in that it provides a single, uniform underlying structure for all components of a system.

An inheritance mechanism is useful in that it simplifies programming. Rather than create a unique type of abstraction for each logical entity the program manipulates, sets of related concepts are created. These are organized hierarchically, with more general concepts (abstractions representing both storage and operators) at the *top* of the hierarchy. A generalized concept is specialized into more specific types by adding or overriding representations and operators. At the lowest level of the hierarchy are the most specific types of individual entities. An example of an inheritance tree (tangle) for some finite element concepts is shown in Figure 2.6.

An actual object may have parents from more than one hierarchy (*tangle* inheritance). This multi-level graphical structure of abstract types lets the programmer structure objects so that shared concepts are not repeated, and lets different concepts remain *orthogonal* rather than being combined into a single piece of code (a detailed example of orthogonality for matrix types and representations is presented below).

Polymorphism is present (in a limited form) in many languages. For example, the "+" operator in FORTRAN is polymorphic in that it can be applied to real, integer or complex variables. Polymorphism consists of using a single operator to represent an operation on a variety of data types (operator *overloading*) while providing the means to differentiate between operators with the same name that are applied to different data types (operator *dispatching*). Object-oriented languages provide the capabilities to define an operator that can be applied to a variety of data types (e.g., define "+" for reals, integers, time, matrices, linked-lists), and to automatically invoke (at run-time) the correct

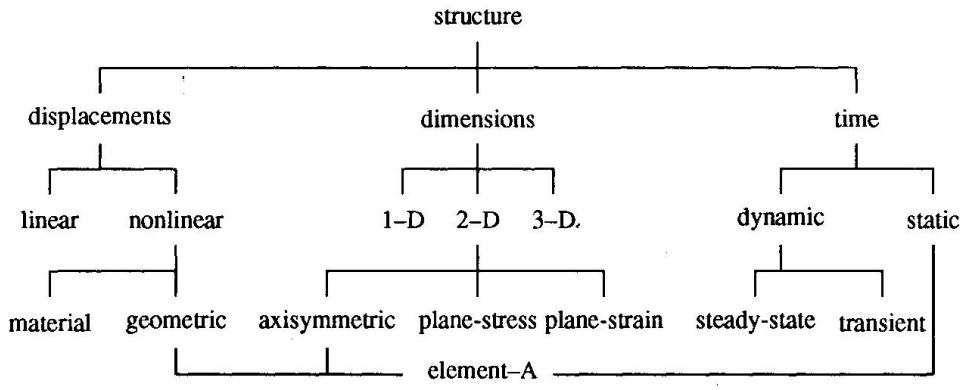


Figure 2: Example Inheritance Tangle

code based on the types of objects to which the operator is applied (i.e., the code used to add two time variables is different from the code used to add two matrices, but this is not apparent from the program).

Together these aspects of object-oriented languages are useful in simplifying programming. Creating abstract data types is natural; type completeness makes programs conceptually "cleaner"; inheritance simplifies organization and promotes reuse of code; and polymorphism can be used to implement constraints and to provide a single notation for programming mathematically similar operations on distinct types.

No other methodology provides all of these characteristics. Using alternative implementation strategies is feasible, but requires more complex programming (essentially their use would require building what are the inherit characteristics of an object-oriented language). The current work uses CLOS (Common Lisp Object System) as the actual implementation language.

3 Solution Approach

The techniques outlined above are being used, individually and together, in developing finite element programs[1,2]. At this time, work is centered on detailed explorations of the proper use of the techniques individually; the integration of the techniques into a unified framework for finite element programming is still under study.

The current target use of the programming environment is for FEM researchers. This community is more demanding in their need for a flexible environment which simplifies programming, and while resource utilization is important, it can be treated as a secondary issue.

The key components of the solution are the use of abstract types and an associated formal specification. Additionally, representation of control and knowledge are important components, as is an approach which is applicable to a parallel hardware base.

The conceptual model of how to integrate these components is centered on a constraint-based or data dependency-based task scheduler. A program is represented as a set of abstract data types. Associated is a representation of fundamental dependency relationships and constraints on the data items. A computational task is selected on the basis of user requests for results. Heuristics are used as needed to select from competing alternatives, and resource information is used to allocate the task to one of many alternative processors in a parallel implementation. As the computations proceed, declarative information from constraints and data dependencies are used to select additional tasks which must be completed to solve the problem.

In this mode of problem solving, the program is dynamically generating the problem-solving strategy. As such, components of the program can be changed or replaced without concern for the implications of the change. A single global program does not exist, but is generated from the available components as needed on the basis of those components available, the knowledge, and the problem characteristics.

The control strategy outlined above has not yet been implemented. In the current implementation, control remains procedural, with imbedded implicit knowledge of problem-solving sequences. While the implemented approach lacks

the elegance one might like, it is still important in that it is derived from a declarative representation, based on abstract data types.

As noted above, most of the techniques described form the basis of how to express problem-solving knowledge, with object-oriented programming being the implementation strategy. What information is represented, both in terms of finite element domain information and problem-solving control information, must be "programmed" (represented) using these techniques to provide the complete programming environment. Formal specifications of the abstract data types are used to provide this information and knowledge independently from the types' programmatic implementation.

Numerous abstract data types are needed to define the finite element method. This work is aimed at providing a framework for program development, and is not attempting to define *all* possible abstract types. Some representative abstraction classes include:

- Engineering concepts—dimensions, units
- Mathematical concepts—vectors, matrices, linear algebra
- Structural concepts—load, displacement, stiffness
- Modeling concepts—nodes, DOF, elements, structures
- Representation concepts—sparseness
- Resource concepts—processors, memory, communications bandwidth
- Finite element concepts—specific element types

These items form a hierarchy of concepts. For example, a *load* can be represented as a vector of values with units of force (i.e., a vector of force objects). Similarly, a structure load matrix can be represented as a matrix of loads (each term is a vector instance of the load type) stored in some representation (full matrix or some sparse representation).

Some examples of these data types and their specifications, along with their use in problem solving, are described below.

3.1 Matrix Abstraction

As an example, consider an abstraction for a matrix data type (a more detailed presentation of this example is contained in [3]). The matrix abstraction must provide a representation and a set of operators. Since in terms of problem solving, matrices with special properties are common (e.g., symmetric, triangular), the abstract type is divided into classes for the various type of matrices.

Independent of the class, the same set of operators must be available for all types of matrices (*completeness* is essential so type changes have no other impact). The types of operators provided include: *constructors* (used to create and build matrix objects), *observers* and *mutators* (used to access and change elements of a matrix), coercion (used to change the class of a matrix), mathematics (addition, transposition, multiplication, etc.), control (looping over all elements and *mapping*), and utility operators (direct copy, I/O).

An example of the specification of one operator, solution of simultaneous equations, is:

```
solve = proc (A: matrix[float], C: vector[float], n: integer) returns (B: vector[float])
    requires nrows(A) = ncols(A) = length(C) and A is non-singular and n > 0.
    effects Returns B such that length(B) = min(n,nrows(A)) and A B = C for indices
        0 ≤ i < min(n,nrows(A)).
```

This specification only describes the behavior of the operator: that the equations must satisfy **A B = C**. The actual implementation can decide how to solve the equation (direct, relaxation), or in fact if the implementation is imperative instead of simply a constraint used by a constraint satisfaction procedure.

A number of classes of matrices can be envisioned (general, symmetric, upper or lower triangular, diagonal, square, singular, etc.); the classes are not mutually exclusive. What is important is that the operators can be consistently applied to all instances of a class. For some classes, this implies error detection (inversion should not be attempted on a matrix which is known to be singular). For others, special action by the operators are required (an accessor trying to return a value from the upper triangle of a lower triangular matrix should return a zero). Insuring proper behavior in all situations improves program reliability, and lets the programmer concentrate on the more important aspects of problem solving, not on the details of matrix representation and manipulation.

Independent of the class of matrix is the storage of the matrix elements. Typical examples of storage include full matrices, packed representations (a triangular matrix stored in a vector), linked representations, hypermatrices, banded and skyline. Any class can be stored in any representation. In selecting a representation, the programmer must consider

several issues: access efficiency, storage efficiency, and flexibility of modifying terms. An example of this orthogonality of class and representation is shown in Figure 3.

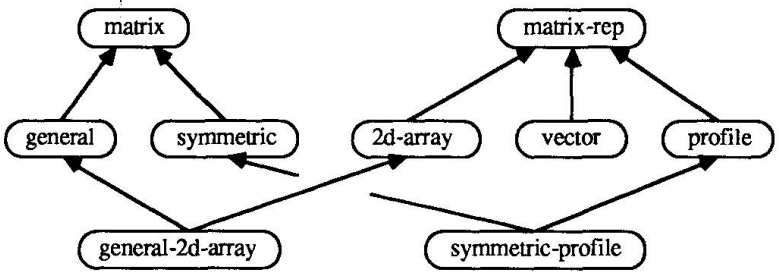


Figure 3: Orthogonality of Matrix Class and Representation

Clearly delineating the class from the representation lets any combination be used: the program (programmer) is free to choose what is best for the problem at hand. Consistency of the operators implies that these choices can be changed (even dynamically with coercion as needed), without any other changes to the program. Separating specification from implementation lets the programmer *tune* the performance of the program as needed, again without impacting other aspects.

3.2 Solution Abstraction

The representation of a structure or element is a graph, with the graph considered in its mathematical sense. Graph vertices are nodes, and links are elements. A full set of graph abstractions and topological operations are provided to manipulate the representation of structures and elements and to form the basis for problem solving. Given such a representation, it is possible to define a procedure for solving a static, linear-elastic problem.

The general solution procedure is to follow a number of steps. Note that this is imperative, procedural control, and is but one of the many alternative sequencing of the steps. The name of the operator used in each step is given in brackets.

- order the nodes of the graph (Reverse Cuthill-McKee) [rcm]
- determine the degrees of freedom of the structure from the set of nodes in the graph [build-dofs]
- partition the ordered nodes based on the boundary constraints [partition]
- generate nodal indices for the assembled system equations [set-indices]
- create the empty stiffness matrix [make-matrix]
- assemble the element matrices by processing all edges in the graph [transform / assemble-stiffness]
- generate the nodal load vector
- solve the resulting equations [solve]

The procedure described above is the process performed by the static-linear-elastic operator. As noted, the steps performed by the operator are implementation-specific. The translation of the procedure into code follows.

```
node-list := rcm(g)
ordered-dofs := build-dofs(node-list)
dofs, nfree := partition(ordered-dofs, source-prescribed?)
set-indices(dofs)
ndofs := vector-length(dofs)
k := make-matrix(list(ndofs, ndofs), symmetric-matrix-1)
for e in edges(g) do
  transform(e, orientation(global))
  assemble-stiffness(e, k)
f := nodal-sources(dofs)
for i := 0 below ndofs do
  f[i] := source(dof[i])
d := solve(k, f, nfree)
```

The specification of the routines used to implement this operator are shown below.

static-linear-elastic = **proc** (g: graph)
requires ∀ (v : member(v,vertex-names(g)) : ∀ (dof : nodal-dof(get-vertex(v,s)) : (source-prescribed?(dof) or state-prescribed?(dof)) and ¬state-set?(dof))).
modifies g.
effects Normally computes and sets the state of: all elements in g, and the free dofs in g. Signals singular-matrix if the assembled stiffness of s is singular.

rcm = **proc** (g: graph) **returns** (list[node])
modifies all nodes in g.
effects The reverse Cuthill-Mckee algorithm. Orders nodes in g by breadth-first-search, reversing and returning the result. Uses mark, marked?, and clear on nodes (thus modifying them) to determine whether or not they have been visited.

build-dofs = **proc** (l: list[node]) **returns** (vector[dof])
effects Returns a vector of all the dofs defined on the nodes in l.

partition = **proc** (v: vector[type], p: **proc** (t: type) **returns** (bool)) **returns** (v-new: vector[type], n: integer or nil)
modifies v.
effects If ∀ (i : $0 \leq i < \text{length}(v)$: p(v[i])) then return v unchanged and nil, otherwise produce a stable rearrangement of v such that ∀ (i : $0 \leq i < n$: p(v-new[i])) and ∀ (i : $n \leq i < \text{length}(v)$: ¬p(v-new[i])) such that n is the index of the first element not satisfying the predicate p.

set-indices = **proc** (v: vector[dof])
requires ∀ (i : $0 \leq i < \text{length}(v)$: ¬index-set?(v[i])).
modifies all dofs in v.
effects Sets the index of each dof in v such that v[index(dof)] = dof.

nodal-sources = **proc** (v: vector[dof]) **returns** (s: vector[float])
requires ∀ (i : $0 \leq i < \text{length}(v)$: source-prescribed?(v[i]) or state-prescribed?(v[i])) and v is partitioned by *source-prescribed?*
effects A bijection using *source* on each of the elements of v. When ¬source-prescribed?(v[i]), sets s[i] = 0.0.

Each of the components of the specification must be implemented to provide the running program. For example, an implementation of the assemble-stiffness operator in CLOS is:

```
(defun assemble-stiffness (e k)
  (let* ((ke (stiffness e))
         (id (incident-dofs e))
         (n (array-dimension id 0)))
    (dotimes (i n)
      (dotimes (j (1+ i))
        (incf (symref k (index (svref id i)) (index (svref id j)))
              (aref ke i j))))))
```

4 Closure

The work described above is just a portion of that underway. To date, the issues of formal specification and data abstraction based on an object-oriented methodology have formed the kernel of work. The most concrete results have been the formal specification of the data abstractions and the implementation of these in the working system. The role of the specification must be emphasized. It is a clear statement of *what* behavior the program must exhibit. It is completely independent from the actual implementation of the program, and a variety of implementations (ranging from declarative to imperative) can be used to transform the specification into the *how* of problem solving.

To meet the goals of a completely declarative finite element system, a number of other topics must be explored in more detail. These include: (1) the issue of alternative control abstractions (other ways of "how to do it"); (2) parallelism and task scheduling for multi-processor hardware bases; (3) incorporation of resource management in a program while

keeping resource issues orthogonal and uncoupled from other concepts; and (4) use of heuristics (e.g., which equation solver to use when) in control of problem solving. Beyond these tasks, the most challenging component of the remaining work is to combine all of these approaches into a powerful, yet "clean" environment (a limited number of clearly defined and distinct concepts) for finite element programming.

This work has demonstrated that the use of alternative programming methodologies can yield more abstract and declarative finite element programs[1,2]. Using the underlying technology and basic tools developed, the implementation and modification of a program require less effort than using conventional imperative programming methodologies.

As stated, the approach is motivated by and has its roots in knowledge-based systems and artificial intelligence, but it is not AI per se, in the classical sense of a problem solver which behaves as a human. When viewed at an abstract level, however, programs built using these techniques do exhibit *intelligent* behavior. At this abstract level, concepts are represented in a declarative form, and details are hidden. The underlying support mechanism processes these declarative forms to perform problem solving, just as classical inference strategies process knowledge.

In closing, it must be noted that the concepts in such an approach are not limited to finite element programming. Rather, the finite element method provides a rich domain to demonstrate a different approach to the development of numerical problem solvers.

Acknowledgements: This work was supported in part by a U.S. National Science Foundation Presidential Young Investigator Award, Grant number ENG-8451533.

5 References

[1] Baugh, J. W., Jr., *Computational Abstractions for Finite Element Programs*, unpublished Ph.D. Dissertation, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, August 1989.

[2] Rehak, D. R., and Baugh, J. W., "Development of an Intelligent Finite Element System," *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing (AI EDAM)*, 1989, in preparation.

[3] Baugh, J. W., Jr., and Rehak, D. R., "Implementation of a Finite Element Programming System — A Declarative Approach," *Computer Utilization in Structural Engineering*, ASCE Structures Congress '89, San Francisco, CA, ASCE, pp. 91-100, May 1989.

Object-Oriented Representation of Design Standards

Représentation «orienté objet» des standards de conception

Objektorientierte Darstellung einer Entwurfsnorm

James H. GARRETT Jr.
Assistant Professor
University of Illinois
Urbana, IL USA

James Garrett, born in 1961, received his BSCE, MSCE, and Ph.D. from Carnegie Mellon University in Pittsburgh, PA. He joined the faculty at the University of Illinois in 1987 and has been performing research in the areas of standards processing, object-oriented building modeling and neural networks.

SUMMARY

This paper describes an object-oriented standards representation in which the logic and data items of a standard are all represented as objects and the methods for manipulating and using the standard are stored within these objects. By using the object-oriented framework described in this paper, it is possible to build a modular, flexible, and powerful representation of a design standard. The benefits of having this natural and declarative description of a design standard are: it makes the logic of the design standard much more apparent than a pure textual representation, it facilitates the automated checking of design objects against a design standard, and greatly enhances the ability to reason about and apply the requirements of the design standard during computer-aided design.

RESUME

Cet article décrit une représentation «orienté objet» des standards de conception dans laquelle les éléments logiques et les variables d'un standard sont tous les deux représentés sous la forme d'objets contenant par ailleurs la manière de manipuler et d'utiliser ces standards. L'utilisation d'un cadre «orienté objet» tel que celui décrit dans cet article, permet de construire une représentation modulaire, flexible et puissante des standards de conception. Les avantages de l'utilisation de cette description naturelle des standards sont: la logique des standards de conception comparativement à une représentation uniquement textuelle: contrôle automatique facilité des objets vis a vis des standards de conception, importante amélioration des possibilités de raisonner avec des standards de conception et de remplir les exigences définies par ces standards pendant la conception assisté par ordinateur.

ZUSAMMENFASSUNG

Dieser Beitrag beschreibt die objektorientierte Darstellung einer Norm. Die Norm ist in der Form von Objekten gespeichert, welche die Methoden für deren Anwendung enthalten. Dadurch wird es möglich, Entwurfsnormen modular, flexibel und sehr anwendungsfreundlich darzustellen. Es ergeben sich folgende Vorteile: Die Logik der Norm wird wesentlich besser ersichtlich als bei einer reinen Textdarstellung. Durch die dadurch mögliche automatisierte Ueberprüfung von Entwurfsobjekten wird die Anwendung im Computer-Aided-Design stark verbessert.

1. INTRODUCTION

As civil engineers, we are required to design buildings, waste treatment facilities, public transportation systems, etc. that conform to a myriad of design standards, specifications, and codes. Although they may have more to deal with, civil engineers are not alone in having to deal with regulation of the performance of their designs; most professional engineers must verify that their designs meet some collection of performance regulations. The ability to properly use these codes and standards (i.e., to correctly identify applicable code provisions, interpret them, and apply them) takes experienced designers years to develop. Because of the many applicable codes that must be considered and obeyed by an engineer and the dynamic nature of these standards, computer-aided usage of design standards is an extremely important component of computer-aided engineering (CAE). Several researchers are working towards a standards representation and processing environment that will free the engineer from concern with the details of any particular standard by assisting him in designing for, and verifying, conformance with all applicable standard provisions [6, 3, 2, 4, 8]. Such an environment would support creative design but limit solutions to be within the bounds of acceptable practice spelled out in the codes. One might argue that such an environment would in fact overly limit creative design and hence be less desirable as a design environment. However, where applicable codes exist, we have an obligation to ensure that our meet the minimum levels of performance spelled out in those codes. It is for this type of design activity, i.e., that falling under the jurisdiction of an existing code, that this standards representation and processing environment is envisioned to support.

For over 20 years, researchers have been investigating ways to represent and automatically use the information contained in a design standard. Fenves, who was the first to propose the idea of formally representing design standards, represented the logic of the standards as a collection of decision tables, where each decision table was responsible for the evaluation of a data item within the standard [5]. Data items were simply defined as the variables, including the provisions themselves, to which the standard refers within its text. When addressing the issue of automated usage of standards, most researchers have treated this formal standard representation as a passive entity to be acted upon by a single, monolithic standards processor, in much the same way that most pure rule-based systems rely on a single inference engine. A more flexible method of representing and processing a standard, which is the subject of this paper, would be to provide each data item the capability of maintaining its own dependencies, of determining its own value, of retrieving the necessary data from a design description, etc., using an object-oriented approach. In other words, treat each data item identified in the standard as a self-contained object that contains all information particular to that data item and all methods for manipulating that information.

The purpose of this paper is to describe such an object-oriented standards representation and processing environment. The remainder of this paper describes: 1) the basic function and form of a design standard, 2) the general concept of object-oriented programming, and 3) the proposed object-oriented model of a design standard.

2. DESIGN STANDARDS

The basic function of a design standard is to state requirements that must be met in order to ensure that an adequate level of performance for an entity is provided. These requirements are derived from experience with successful and unsuccessful designs. As more and more knowledge is gleaned from design experience and experimental research, the definition of adequate performance, and thus the design standard, are further refined.

Most standards state minimum levels of acceptable performance and identify a collection of criteria that quantitatively define what acceptable performance means. Each criterion is a logical expression of some set of variables, or data items. For example, in Fig. 1. a portion of the AISC LRFD specification is given from which data items and and the criteria can be determined. The requirement is that there be adequate compressive strength for compression members; the criterion for determining that adequate compressive strength is provided is $P_u \leq \phi_c \; P_n$, where P_u is the factored compressive load on the member. This section is predominantly concerned with defining the data item F_{cr} .

The data items within a standard can be classified as follows:

1. *basic* – no explicit expression is provided in the standard that defines its value, hence, it's value is to be retrieved from the design being evaluated or from general knowledge of the domain (e.g., E, r, F_y , K, l);

2. *derived* – an explicit logical or mathematical definition for deriving its value is provided within the text of the standard (e.g., F_{cr} , P_n, λ_c); and

3. *requirement* – a special type of derived data item that identifies the criteria that must be satisfied and has one of the following status values: "satisfied", "violated", or "not applicable" (e.g., design-compressive-strength).

As can be seen from the above described example, precedence relationships exist between the data items within a design standard and the methods to use in determining the value of a data item are many times dependent on the value of other data items.

" The design strength of compression members whose elements have width-thickness ratios less than λ_r of section B5.1 is $\phi_c P_n$

$$\phi_c = 0.85$$
$$P_n = A_g F_{cr} \qquad \text{(E2-1)}$$

For $\lambda_c \leq 1.5$

$$F_{cr} = (0.658^{\lambda_c^2}) F_y \qquad \text{(E2-2)}$$

For $\lambda_c > 1.5$

$$F_{cr} = [\frac{0.877}{\lambda_c^2}] F_y \qquad \text{(E2-3)}$$

where

$$\lambda_c = \frac{Kl}{r\pi}\sqrt{\frac{F_y}{E}} \qquad \text{(E2-4)}$$

A_g = gross area of member, $in.^2$
K = effective length factor
l = unbraced length of member, in.
r = governing radius of gyration about plane of buckling, in.

For members whose elements do not meet the requirements of Sect. B5.1, see Appendix B5.3. "

Figure 1. — Excerpt for AISC LRFD [1] - Chapter E, Section E2

3. OBJECT-ORIENTED METHODOLOGY

The basic building block of an object-oriented representation is the *object* — a modular, self-contained collection of descriptive attributes and the procedural methods for manipulating those attributes. Representation in an object-oriented environment first requires the description (declaration of attributes and methods) of the general types of objects that populate the domain (class objects), and then requires the generation of instances of the class objects to describe the particular entity being modelled.

In object-oriented representations, everything is an object. Objects can represent concepts, physical objects, processes, etc. In all cases, the object possesses a set of attributes and methods. Attributes represent data about the object; methods represent processes that the object is capable of performing. Attributes and methods are usually both represented as *slots* within the object. Other objects can access these slots, but only by sending a message to the object that "owns" the data or method. In addition to having a value, the attributes of an object may also have self-descriptive information, such as permissible range or type. This

information is stored in *facets* that are associated with the slots. A special type of facet, called a procedural attachment or demon, watches a slot value and executes a method when that value is added, changed, or erased. This feature of object-oriented environments is especially suited for performing event-driven computation and will be extensively used in the object-oriented modelling and usage of standards.

Fig. 2. shows the typical structure and an example of a data-item object. In that example, the first four slots store declarative information about the data-item, such as its value or its ingredients. The "value", "ingredients", and "dependents" slot all have facets that describe what to do if a slot value is needed or erased.

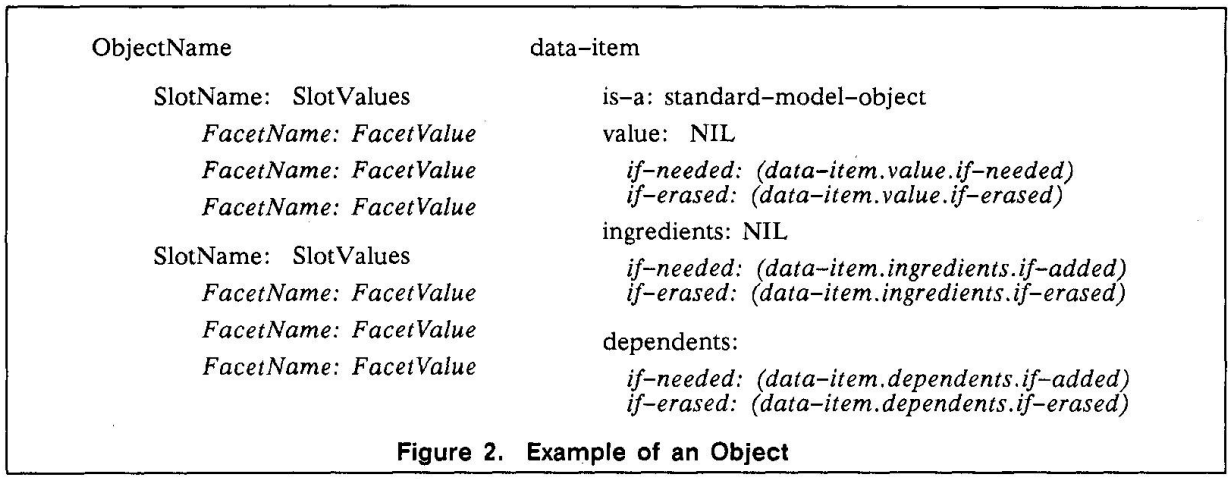


Figure 2. Example of an Object

A common practice in object-oriented programming is to develop templates for types of objects, commonly called *class* objects. These class objects usually possess attributes, attribute values, method names, or methods that are common to several more specific objects. If these more specific objects are themselves templates for other even more specific objects, they are called *subclass* objects. Objects that represent an specific instance of a class or subclass object are called *instances*. Instances are *children* of subclasses (or classes), subclasses are children of classes (or other subclasses), classes are *parents* of their subclasses and instances and subclasses are parents of their instances or other subclasses. These parent-child relations are important because in most object-oriented programming environments, children automatically *inherit* attributes and methods from their parents. For example, all instances of the object data-item (shown in Fig. 2.) will inherit the slot names "value", "ingredients" and "dependents", and their procedural attachments from the object data-item. Through inheritance, it is possible to represent information at an appropriate level of object generality and have all more specific instances of objects inherit that information, thus reducing redundancy and improving consistency.

Hence, the key ideas of object-oriented programming are that objects possess attributes and methods, can inherit attributes and methods from other objects, and communicate with each other (i.e., get data or execute an object's method) only by sending messages.

4. OBJECT-ORIENTED DESIGN STANDARD MODEL

As was stated previously, the purpose of this paper is to describe an object-oriented model of a design standard that facilitates automated standard conformance verification. In order to ba able to fully automate the verificalion of a design for conformance with applicable design standards the following are necessary:

1. an object-oriented model of the data items (both basic and derived) to which the standard refers and the logic for determining the value of each derived data item expressed in the standard;

2. an object-oriented description of the entities to which the standard applies, which represents the attributes of each entity within the scope of the standard and serves as a repository of knowledge about the entities not found in the design standards; and

3. a collection of mappings: 1) between the basic data items in the standard and the attributes of the design description, and 2) between design description objects and behavior limitations (see Section 4.3.) in the standard model.

All three parts of the model are required in order to ensure proper automated interpretation, not just the first. Most standards processors have provided the first part and a little of the second in the form of a hierarchy of classifiers [6]. Elam's and Lopez's work identified the need for, and implemented in limited form, all three parts [4]. This work presents an architecture, cast in an object-oriented framework, that includes a representation of standard logic (explicit knowledge contained in a standard), the underlying description of the entities within the scope of the standard (implicit knowledge in some standards), and the mappings between data items in the standard and the attributes of the design description.

4.1. Object-Oriented Design Standard Logic Model

4.1.1. Objects in the Standard Logic Model

As illustrated in Section 2., a design standard is a collection of logically interrelated data items. These data items and their logical interrelationships are represented using the following object classes, the hierarchy of which is shown in Fig. 3.

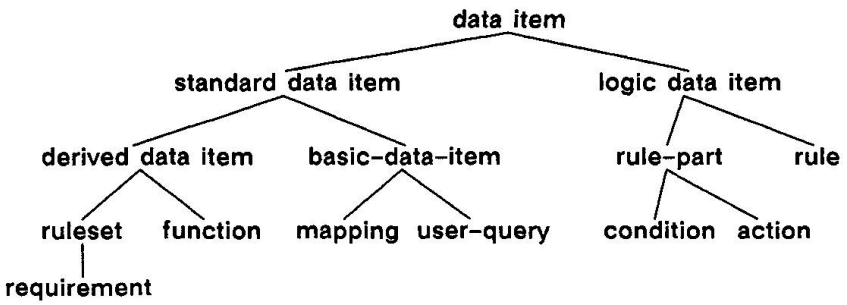


Figure 3. — Hierarchy of Object Classes for Standard Logic

Data item. This class of objects is the most general class in the hierarchy describing the general properties of data items used to represent a standard. A data item is defined to be an explicitly represented variable that has the following properties: a value (either computed or input by the user), a list of ingredient data items upon which their computed value is based, a method (ruleset, function, mapping, or prompt-string) for determining its value from the list of ingredients, and a list of data items that depend on its value. The hierarchy in Fig 3. shows two subclasses of data-item: standard data item and logic data item.

Standard Data Item. This class of objects represents what can be thought of as the "traditional data items" of standard representations — the data items referred to explicitly within the text of the standard. Standard data items are specialized according to the method in which their value is determined: derived or basic.

Derived Data Item. This class of objects represents the data items referred to explicitly, and given a method of evaluation, within the text of the standard. Derived data items are specialized according to the method in which their value is computed. Other subclasses of derived data items may be added later because of the flexibility of this object-oriented approach, but for now only functions and rulesets are defined.

Ruleset. This class of object represents a specific type of derived data item whose value is conditional and whose evaluation strategy is thus represented as a collection of rules. A ruleset is almost identical to a decision table in that all of its rules focus on the evaluation of a single data item. The ruleset offers a little more flexibility in that the inference strategy may be varied; this flexibility gained in using rulesets over decision tables was promoted by Elam and Lopez [4]. Like all data items, rulesets have ingredients. But, unlike past representations, the ingredients to a ruleset are the rules that make up the ruleset (i.e., logic data items), not standard data items.

Requirement. This class of objects is a special subclass of ruleset that describes a requirement with a set of criteria (or conditions) from which its value ("satisfied", "violated", or "not applicable") is computed. Requirements are also instances of behavior limitation objects, which are described in Section 4.3.

Function. This class of objects represents a second subclass of derived data item, whose evaluation method is a non-conditional function, similar to an action of a ruleset. Like all data items, functions have ingredients which are the data items appearing in the expression of the function.

Basic Data Item. This class of objects represents the data items referred to explicitly within the text of the standard, but not given a method of evaluation within that text. For these data items, it is assumed that the user will provide the needed value either from a design model or directly in the form of an answer to a query. Hence, basic data items are specialized according to the method in which their value is retrieved: mapping or user-query.

Mapping. This class of objects represents a subclass of basic data items. A mapping is a declarative description of where to look in, or how to compute a value from, the object-oriented design description. Mappings are described in more detail in Section 4.3.

User Query. This class of objects represents a second subclass of basic data items. A user query data item simply describes a data item for which it is known a priori that its value will have to be asked for from the user. Because of this a priori knowledge, the user query object contains a prompt-string to use in querying the user, type and range information for checking user input, and default values in case the user does not know the answer but wishes to continue.

Logic Data Item. — This class of objects, a subclass of the data item object, represents such items as conditions, actions or rules, that are used to describe the logical relationships between standard data items. Each logic data item possesses an evaluation method. For conditions and actions (rule-parts) , this evaluation method is in the form of an algebraic expression. For rules, this evaluation method is in the form of a list of condition-value pairs and an action to perform given those conditions match those expressed for the rule. Layers of standard data items are related through layers of logic data items. By having these logic data items explicitly represented, it is possible to maintain much more refined dependency relationships. Past representations would invalidate a data item if any of its ingredient data items changed, which may not have been necessary if the rule used to originally compute the dependent data item did not depend on that ingredient.

Rule Part. This class of objects represents conditions and actions, which have a symbolic expression for their description of evaluation strategy.

Condition. This class of objects has as its evaluation method an algebraic expression that evaluates to either T or F. The ingredients to a condition are defined to be the data items contained within the symbolic expression of the condition; the dependents of a condition are the rules that refer to that condition.

Action. This class of objects has as its evaluation method an algebraic expression with no restriction on its value. The ingredients to an action are defined to be the data items contained within the symbolic expression of the action; the dependents of an action are the rules that refer to that action.

Rule. This class of objects possesses an attribute for storing a pattern of condition-value pairs and an action to execute in the event that the condition-value pattern matches the actual condition-value situation. The ingredients to a rule are the conditions and action to which the rule refers; the dependent of a rule is the ruleset (a derived data item) to which the rule belongs.

4.1.2. Example Standard Logic Model

To illustrate the use of the above described objects in modeling and evaluating the logic of a design standard, Sect. E.2 from Chap. E of the American Institute of Steel Construction Load and Resistance Factor Design Specification [1] (see Fig. 1.) is modeled using the above described objects (see Fig. 5.).

To determine the value of DESIGN_COMPRESSIVE_STRENGTH, it must be sent a message to return its current value. The DESIGN_COMPRESSIVE_STRENGTH object, being an instance of a ruleset, responds to the message by sending messages to the rules in its RULES slot: DCS-1, DCS-2 and DCS-3. The ruleset first sends a message to rule DCS-1; if DCS-1 does not respond with a non-NIL value, the next rule is messaged. DCS-1, being a rule, responds to a message for its value by sending messages to each of its

identified conditions and then checks if the condition returns the value indicated for the rule. For example, DCS-1, in response to a message for its value, sends a message to the object LOCAL_BUCKLING_SATISFIED to return its value and if this object responds with a value "T", DCS-1 then sends a message to the object DCS_E2_SATISFIED to return its value.

When a condition is defined, i.e., its symbolic expression is filled in, this expression is parsed to determine the ingredients and is transformed into a LISP-evaluatable expression. Hence, when a condition is sent a message to return its value, it sends messages the identified ingredients and then evaluates the LISP-evaluatable expression with the returned ingredient values. Thus, when DCS_E2_SATISFIED is sent a message, it responds by sending messages to the ingredients Pu, PHI-C, and Pn.

Pn, being a function, responds by sending messages to its ingredients and evaluating its LISP-evaluatable expression, both of which were generated when the symbolic description was defined. Hence, when a function is messaged, it sends messages to its ingredients for their values and then evaluates the LISP-evaluatable expression with the returned ingredient values. When Pn is sent a message, it sends messages to Ag and Fcr to return their values. Fcr, being a ruleset, reacts to a message to return its value exactly as the DESIGN_COMPRESSIVE_STRENGTH ruleset did. This recursive process of messaging rules, conditions, actions and functions continues until the objects receiving messages are instances of the basic-data-item class, such as Ag, Kx, Lx, etc.

There are two types of basic data items, user-queries and mappings. When a user-query is messaged, it simply prints out its prompt string and checks the users response against type and range information stored in the query object. When a mapping is sent a message, it retrieves the information from the object-oriented design description (described in Section 4.3.). After the values of these basic data items are retrieved and backpropagated to the derived data items, their values can be computed. After the values for these derived data items have been backpropagated to logic data items, their values can be computed. And finally, after the values of these logic data items have been backpropagated to the requirements, their values can be computed.

4.1.3. The Benefits of an Object-Oriented Representation of Standard Logic

Because the decision table for so long has been the main way of representing the logic within a design standard, one must ask why this object-oriented model is any better. The benefits of using this object-oriented approach all basically relate to flexibility and are described as follows.

1. Every object maintains its own strategy for determining its value. In other words, there is no central definition of what a condition is, of what an action is, of what a rule is, etc. Although the predominant kind of function is the algebraic expression in terms of other data items, this does not have to be the only kind of function. For example, a subclass of function could be defined to be a neural network that has been trained to recognize a collection of data that defies mathematical description, which when given a set of ingredient values as input, returns a value for the function. Similarly, one could have more than one type of ruleset (fire-one rule, fire-all-rules, etc.), condition (symbolic, numeric, neural), and rule (AND rules, OR rules, etc). Such flexibility can only be achieved when the representation is object-oriented, where the object requesting a value need not know with what kind of object it is dealing.

2. The values and ingredients for individual conditions and actions can be maintained individually, whereas for a decision table all ingredients for all conditions and actions are lumped into the set of ingredients for the data item evaluated by the decision table. This lumping of ingredients causes data items to be unnecessarily nullified when an ingredient's value is changed. If the ingredient was not actually used in the evaluation of a data item, its change should not invalidate the dependent data item.

3. By also representing basic data items as objects, a place is provided to store data item specific prompt strings, range and type information, and mappings. The storing of this information with the basic data items makes for a much more organized and flexible description of a design standard.

4.2. Object-oriented Design Description

The design description is intended to perform many functions. First, it serves to identify the entities to which the standard requirements are intended to apply. Second, this model will identify, for each design object, a set of attributes from which the basic data items found within the standard can be computed. This will permit the generalized expression of the mappings between the data items of a standard and the attributes of these design entities. The design descriptions can then be mapped to a much larger, global model using the knowledge-based database interface mechanisms of KADBASE [7], or a similar data communication environment. In fact, because such a global model does not yet exist, it is most likely that the design descriptions developed for various standards will play a part in determining the information content of such a global model.

4.3. Mappings Between Standard Model And Design Description

There are two types of mappings between the standard data items and the design description objects: 1) the mappings between the design description objects and the various applicable requirements of the standard, and 2) the mappings between the basic data items in the standard model and the design description object attributes. The first set of mappings is essentially used to determine which of the standard requirements are applicable for the design entity in question and need to be checked. This mapping is simply represented as a slot in the design description objects which contains a list of applicable behavior limitation objects [6] (in the structural case - behavior limitation objects represent combinations of stress state and limit state). Each of the requirement data items in the standard model is an instance of the behavior limitation object to which it applies. Thus, the layer of behavior limitation objects is the medium of communication between a design description and a standard; these behavior limitation obejcts are the components of the classification system used in previous standards processing environments [6].

The second set of mappings go in the opposite direction to the previous set by linking data items to the appropriate slots of the objects in the design description. The concept of this type of mapping was proposed and implemented by Elam and Lopez [4]. However, their mappings were hardcoded queries into a specific database. The mapping concept envisioned here is similar to that of Elam and Lopez, but instead of expressing the mapping in terms of a query, it is expressed in the form of a set of attribute names (present within the object-oriented design description) and a symbolical expression for combining those ingredients into a value for the basic data item. For example, consider the basic data item rx, the radius of gyration about the x axis, shown in Fig. 4. When the value of rx is requested by some other derived or logic data item, the mapping object responds to that message in the following manner.

1. The mapping object sends a message to an object, called the "context object", to return the name of the design description object for which the standard is being checked.

2. The mapping object then sends a message to this design description object, requesting values for the attributes named in the "DESIGN-ATTRIBUTES" slot of the mapping object.

3. The mapping object then evaluates the symbolic expression, with the given attribute values, to determine the value of the mapping data item.

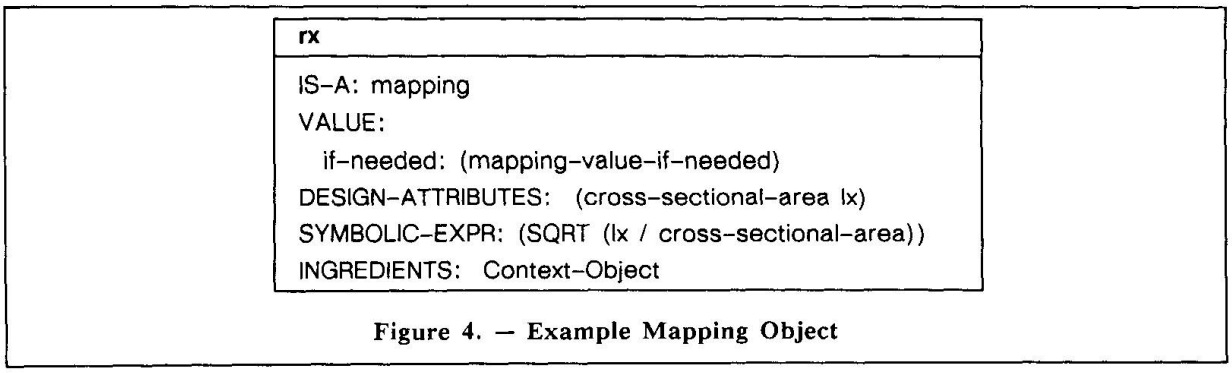
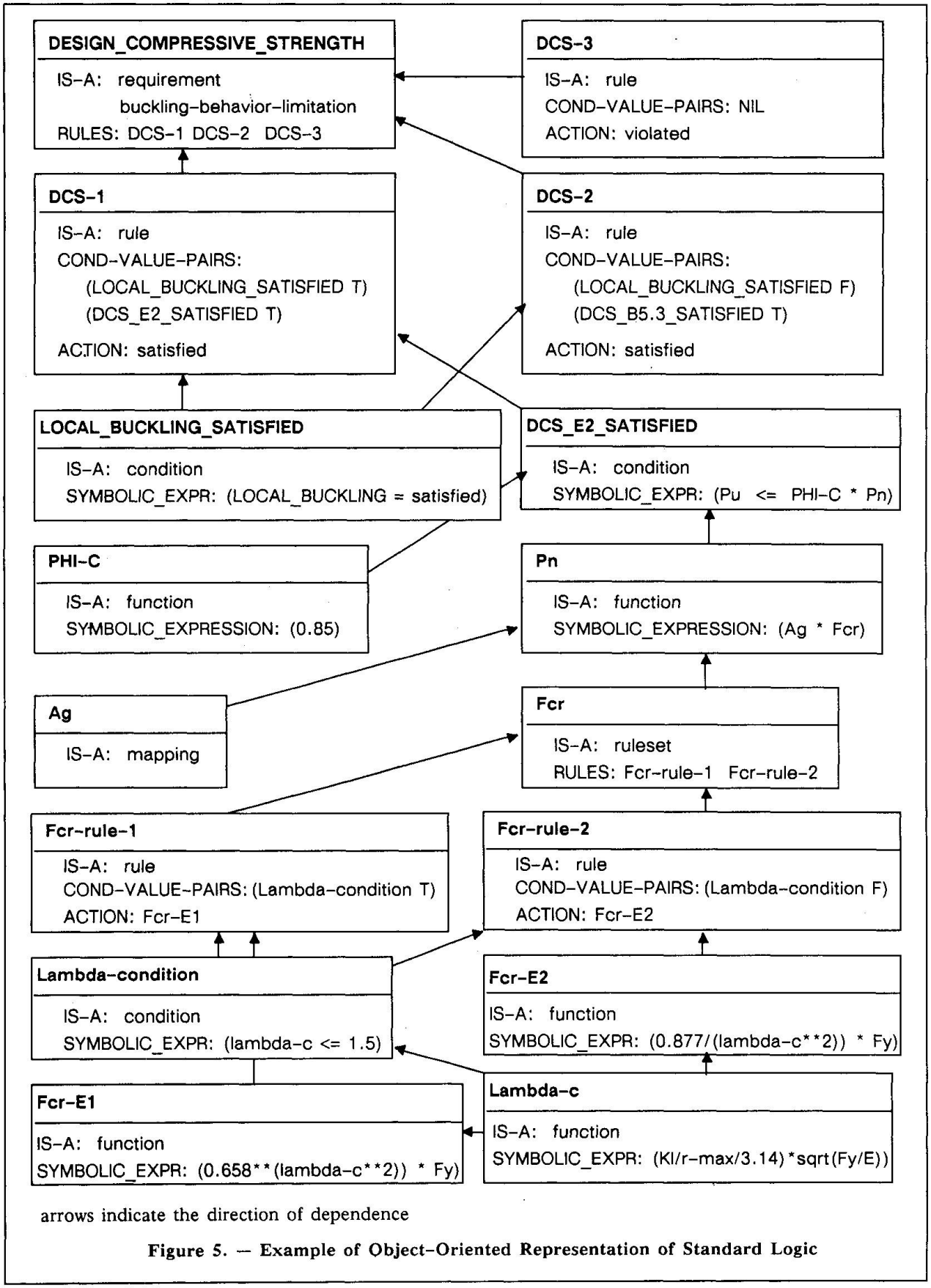


Figure 4. — Example Mapping Object



arrows indicate the direction of dependence

Figure 5. — Example of Object-Oriented Representation of Standard Logic

Note, that the expressiveness of the mappings is dependent on the attributes present within the design description. If the attributes needed to compute a basic data item are not present or available for reference, the default behavior is to query the user for the design description attributes. Also note that initially it has been assumed that all of the information needed for the determination of a basic data item can be found in the design description object that initiated the checking of the requirement. However, it is planned to extend this assumption as follows: the information needed is either present within the design description object or in some other object that is explicitly related to this object (i.e., by a part-of or connected-to relationship).

5. CONCLUSIONS

This paper describes an object-oriented representation of a design standard that provides more flexibility in the representation and usage of the information present within a design standard. All information within the design standard is represented as instances of predefined object classes. The benefits of having this declarative, object-oriented description of a design standard are: 1) it facilitates the automated checking of a design for conformance with a design standard, 2) it facilitates the ability to reason with and manipulate the requirements of a design standard during computer-aided design, and 3) it has the representational flexibility of an object-oriented approach. In addition, by having the design entities to which the standard applies represented as part of the model of a standard, it is possible to describe a set of mappings between the standard and this design description that will ensure proper interpretation of the basic data items in the standard, as well as the derived and requirement data items.

6. ACKNOWLEDGEMENTS

This material is based on work supported by the National Science Foundation under Grant No. DMC-8808132. The Government has certain rights to this material.

7. REFERENCES

1. *Load and Resistance Factor Design Specification for Structural Steel Buildings*, Amercan Institute of Steel Construction, Chicago, IL, 1986.

2. CRONEMBOLD, J. R. and K. H. LAW, "Automated Processing of Design Standards", *Journal of Computing in Civil Engineering*, Volume 2, Number 3, pages 255-273, July, 1988.

3. DYM, C. L., R. P. HENCHEY, E. A. DELIS and S. GONICK, "A Knowledge-Based System for Automated Architectural Code Checking", *Computer-Aided Design Journal*, Volume 20, Number 3, April, 1988, pp 137-145.

4. ELAM, S. L. and L. A. LOPEZ, "Knowledge Based Approach to Checking Designs for Conformance with Standards", Technical Report CESLRS No. 9, Department of Civil Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 1988.

5. FENVES, S. J., "Tabular Decision Logic for Structural Design", *Journal of the Structural Division*, Volume 92, Number ST6, pages 473-490, June, 1966.

6. GARRETT, JR., J. H. and S. J. FENVES, "A Knowledge-Based Standard Processor for Structural Component Design", *Engineering with Computers*, Volume 2, Number 4, pages 219-238, 1987.

7. HOWARD, H. C., "KADBASE: An Expert System/Database Interface", in proceedings of The Fifth Conference on Computing in Civil Engineering, pp. 11-32, March 1988.

8. RASDORF, W. J. and T. E. WANG, "Generic Design Standards Processing in an Expert System Environment", *Journal of Computing in Civil Engineering*, Volume 2, Number 1, pages 68-87, January, 1988.

Expert Systems for Engineering Codes

Systèmes experts pour les normes en génie civil

Experten-Systeme für Bauingenieurnormen

Ron SHARPE
Senjor Princ. Res. Scientist
CSIRO
Melbourne, Australia

Ron Sharpe, born 1943, got his PhD in civil engineering degree at Southampton University after graduating from Melbourne University. He now leads a knowledge based systems research group.

Bertil MARKSJÖ
Princ. Res. Scientist
CSIRO
Melbourne, Australia

Bertil Marksjö, born 1939, graduated from the Royal Institute of Technology in Stockholm and followed up with a PhD. He is deputy leader in the same knowledge based systems research group.

SUMMARY
Both users and developers of complex engineering codes have much to gain from the use of expert systems to help ensure correct interpretation and avoidance of obscurities, contradictions and omissions. The paper discusses the development of PC based expert systems for the Australian wind and building codes including the problems encountered. The former system is complete and being introduced into design offices.

RESUME
Dans le cas des normes en genie civil, l'utilisation d'un système expert peut être profitable aussi bien aux utilisateurs qu'au personnes en charge de les établir, en les aidant à garantir une interprétation correcte du code et à éviter les contradictions, les omissions ou les parties obscures. Ce document discute les développements de deux systèmes experts sur PC, l'un étant lié aux standards australiens du bâtiment, le second à ceux des effets du vent sur les édifices. Le développement de ce dernier a maintenant été complété et le système est actuellement présenté à des bureaux d'études.

ZUSAMMENFASSUNG
Um Bauingenieurnormen besser zu interpretieren und Unklarheiten und Widersprüche zu vermeiden, können Experten-Systeme eine wichtige Rolle spielen, sowohl bei den Benutzern als auch bei den Sachverständigen selber. In dieser Veröffentlichung wird die Entwicklung von Experten-Systemen für die australischen Wind und Baunormen beschrieben, einschliesslich der dabei auftretenden Probleme. Das WINDLOADER-System wird zur Zeit in Ingenieurbüros eingeführt.

1. INTRODUCTION

The problems in developing codes are well known [1] and expert systems can reduce these as well as assist users [2,3]. For the developer they encourage greater precision in code specification, and exposure of inconsistencies, omissions and ambiguities. For the user they offer accurate and thorough checking of relevant clauses faster than possible by hand, plus an ability to explore more design alternatives.

In spite of these benefits few expert systems for codes (and also for engineering in general) have been completed and implemented even though many small prototypes have been developed in academic and research institutions. One of the reasons is that the development of full-scale systems usually requires extensive resources well beyond that required for the initial prototype and these are usually not available. Also many engineering systems combine logic with mathematical, database and graphical procedures. While expert systems are designed to handle logic, they also need to be integrated with the other procedures in order to achieve wider usage [4].

2. WINDLOADER

2.1 History of WINDLOADER

Over the past five years WINDLOADER has been developed in parallel with the latest revision of the Australian wind loading code, AS 1170.2 [5]. The code is complex and a study [6] has shown that designers can make significant errors in its interpretation. This code was previously revised in 1975 and 1983, and the latest version represents a major revision involving a Simplified Procedure for small buildings less than 15 metres in height, and Detailed Procedures for both static and dynamic analysis. WINDLOADER is restricted to the Detailed Static Analysis section since this is where most code users are likely to need assistance. (WINDLOADER may be extended to cover dynamic analysis in the future if there is sufficient demand.) Most of the static analysis section has also been incorporated into the New Zealand loading code, and it is expected that a version of WINDLOADER will be developed for that country.

The complexity of the code has increased substantially in the latest version and users are expected to experience much difficulty, especially in complicated applications, in the next year or so with the text version of the code. A series of seminars and a code commentary have been prepared to assist in the transition period, and WINDLOADER is also expected to greatly assist users.

The development of WINDLOADER began as a prototype system [7] coded in Melbourne University Prolog in 1985 on a HP9000/540 computer, followed by conversion to Prolog-2 on an IBM PC-AT computer with a Professional Graphics Display in 1986. The prototype (which featured colour graphic menus and displays) generated much interest and proved very useful in attracting both funding support and involvement of experts in the development of a commercial system in 1987. At this stage Standards Australia conducted a survey of potential users and found that 85% of potential users had access to an IBM PC or compatible computer, and so it was decided that WINDLOADER should be developed for the PC market. However this meant that Prolog alone would be unsuitable for a system as large as WINDLOADER since it would not be capable of fitting into a PC. It had also been found during the prototype development that Prolog was too slow for recursion and too cumbersome for processing graphics, equations and table interpolations.

In 1987 it was decided to develop an in-house shell (called BX-Shell) written in C and use this for WINDLOADER since no suitable commercial PC shell could be found at that time with the necessary range of functions, speed and low delivery cost. The BX-Shell included special C functions for

processing graphics, equations and tables [8]. The shell also included BX-Prolog for handling logic. BX-Shell was developed on a Sun 3/50 workstation with the intention of later porting it down for use on PCs.

2.2 Use of Crystal Shell

In early 1988 the Crystal PC shell became available and was quickly adopted in place of the BX-Shell since Crystal met most of the desired development and delivery features required. Crystal is written in C and while it did not possess all of the functions required by WINDLOADER for equations and table handling, it was possible to quickly port these across from the BX-Shell.

Crystal is a rule-based shell and this suited the format of the wind code which is also partly rule based.

2.3 Knowledge Representation

While the printed code is intended to be exhaustive and self-explanatory, development of the expert system required that the expert had to be consulted several hours per week for code interpretations over the development period. The printed code deals with a complex domain and even experts may need to spend many hours to ensure that correct interpretations of difficult sections are made for the full range of possible cases.

The wind code is partly rule based and partly procedural. The procedural sections include extensive use of mathematical equations and table interpolations, many of which are non-linear and not continuous. Such procedural sections are often tightly interwoven into the code logic and it is frequently very difficult to encode the knowledge into an easily readable format. This makes it difficult to achieve a close correlation between the expert knowledge (the printed code) and the knowledge representation in the expert system.

For example, consider the following subset of clauses from Section 3.2.6 for determining changes in terrain category:

> *Fully developed gust windspeed multipliers (Mz,cat) only apply at a structure site when the terrain category at the site is uniform upstream for a distance greater than (2500 + xi) metres.*
>
> *When the immediate upwind terrain extent is less than (2500 + xi) metres, corrected windspeed multipliers (Mx) shall be computed using Equation 3.2.6(3).*
>
> *Notwithstanding this requirement, the extent of upwind terrain to be considered need not exceed the larger of either 2500 or 50 times the structure height (ht), provided that the terrain at that limit is Terrain Category 3 or less rough, (assume the windspeed multiplier (Mo) to be the value for fully developed terrain at that limit).*
>
> *If the terrain at that point is rougher than Terrain Category 3, the upwind limit shall be extended until Terrain Category 3 or terrain of less roughness is encountered, or alternatively fully developed Terrain Category 3 may be arbitrarily assumed upwind of that point.*

The logic of these clauses is complex and first it is necessary to simplify the structure in terms of logical operators (IF, THEN, NOT, etc.) and subclauses (A,B,C, etc.) before rearranging it for the expert system. This gives:

THEN.....A.. IF .....B....
IF .....C.. THEN .....D....
NOT WITHSTANDING ....C...,
THEN .....E... OR .....F...,
IF .....G... OR .....H....
IF NOT(. .G... OR .....H..),
THEN .....I.... OR .....K... OR ...L...

The terms 'NOT WITHSTANDING' and 'NOT(...)' can be removed and the qualifying 'IF....' statements are best placed ahead of 'THEN ...' statements to avoid unnecessary evaluation of the latter if the 'IF...' tests fail. This leads to a simpler and more consistent form:

IF .....B..... THEN ....A....
IF .....C..... THEN ....D....
IF .....G..... OR .....H...,
THEN .....E.... OR .....F...,
ELSE ....I..... OR .....K... OR ...L...

Further changes are required when the subclauses are expanded. Also while it is not stated, a recursive procedure is implied in the last clause (to see if the building penetrates more than one layer arising from upstream changes in terrain roughness). After recursion is included, the logic may be represented in pseudo code as shown in Figure 1. This can be accessed by the user as a WINDLOADER 'help screen'. Finally the actual coding of the clauses in Crystal is different again because of the need to evaluate equations and to store results speedily and efficiently.

```
                ASSUMPTIONS BEHIND CHANGES IN TERRAIN CATEGORY
                                        refs  3.2.6 and E3.2.6

- IF upstream terrain category, Cat, is fully developed at height, Z m
  THEN use multiplier M(Cat,Z).

- IF upstream terrain category, Cat, is undeveloped at height, Z m
  THEN find next terrain further upwind,
     AND interpolate between the upstream multipliers.

- IF next terrain further upwind is undeveloped,
     AND this terrain change occurs beyond point D
         located max(2500,50*Structure height) m from Structure,
     AND terrain category is 3 or less rough at D
  THEN assume a fully developed terrain.

- IF next terrain further upwind is undeveloped,
     AND this terrain change occurs beyond point D,
     AND terrain category is more rough than 3 at D
  THEN find next terrain further upwind until category is 3 or less rough,
     OR arbitrarily assume a fully developed category 3 upwind.

                 | ↑  |  Sections ••                 | ↓  |              |End     |
                 |PgUp|  Pages    ••                 |PgDn|              |to Quit|
```

Fig. 1. Changes in Terrain Category HELP screen.
(This and the following images have been dumped using the PrtSc/PrintScreen key to a dot matrix printer. The color and highlight information is not shown).

The code developers anticipated difficulty would be experienced in this and other sections, and subsequently produced examples in a code commentary which were checked by WINDLOADER.

2.4 User Interface

The design of the user interface required much thought and experimentation. Many of the earlier PC expert shells tended to have either scrolling screens or a sequence of screen images with questions and explanations. Often such systems offer the user little control over the order in which the knowledge base is accessed. In deep systems (with many levels of rules), it is easy for the user to lose his overview and become confused by the order of questions. Later shells have introduced menu formats and pop-up windows and these will become more widely available soon.

A menu-based approach is adopted in WINDLOADER. This appears particularly suited to the design environment where users need as much flexibility as possible to test and modify building shapes, location, orientation and so on. Menus are useful for displaying items for selection, data inputs including tables and results. Users can generally move freely around such menus and select items in any order. A HELP item appears on most menus and this enables one or more explanatory screens to be displayed if needed. These are illustrated in the example of the effect of terrain changes on a 50 m building. The building is in a central business district area surrounded by tall buildings (Terrain Category 4) and there are changes to lesser terrain categories in the east and north directions as shown in Figure 2. Further help screens are provided to enhance the explanation of terrain changes including a diagram (Figure 3).

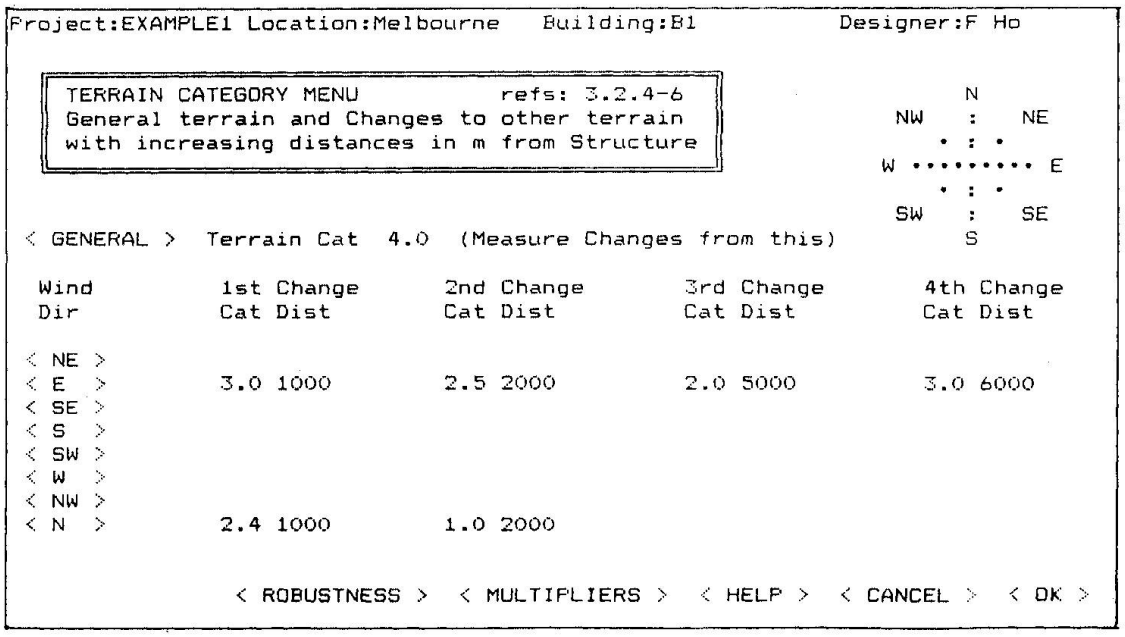


Fig. 2. Input screen for input and checking Terrain Category and changes.

If the MULTIPLIER button is selected in Figure 2, the user will be shown the factors by which the basic wind speeds will be multiplied as a result of the terrain changes. Figure 4 shows that while the protection offered by surrounding buildings in the central business district reduces the basic wind gust speeds by 10 per cent (i.e. to 0.9) at the top of the building, the exposure to the east and north increases the speeds by 7 and 17 per cent respectively. The user can experiment to see how many terrain changes need be considered before the changes in the multipliers become insignificant. The ROBUSTNESS button provides further information on the relative effect of each terrain change. The user can also test the changes in building height and location if wind loads need to be reduced to achieve cost savings or increased safety, especially in cyclone areas.

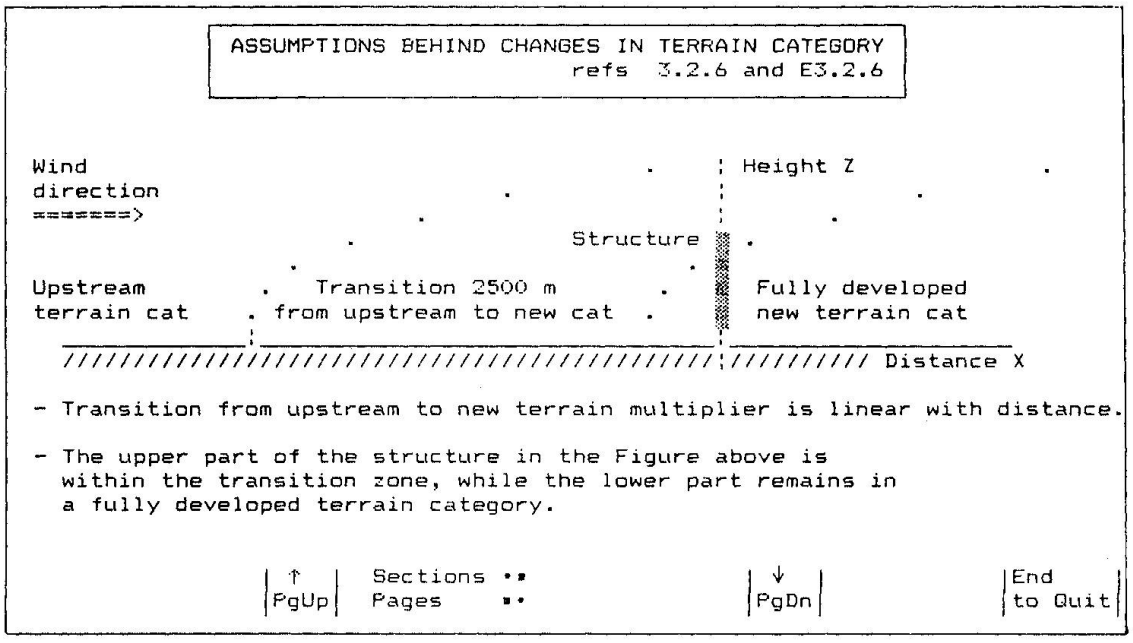


Fig. 3. HELP screen for diagrammatic explanation of terrain change effects on a structure.

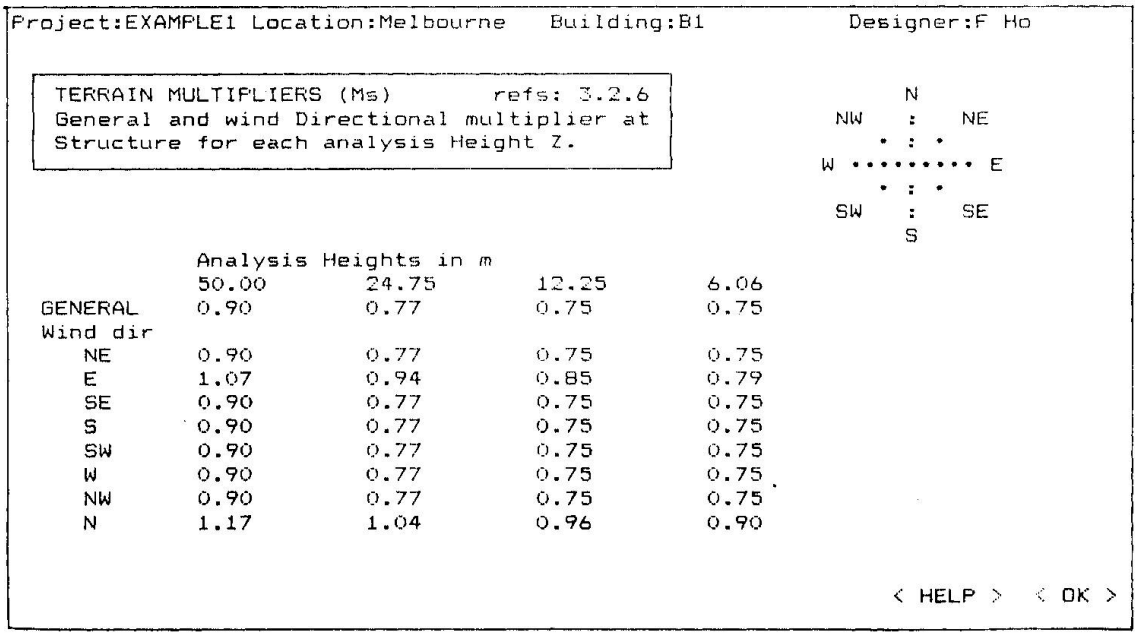


Fig. 4. MULTIPLIERS explanation screen showing impact of terrain changes in east and north directions.

Most of the user interface tends to be procedural as in algorithmic software systems including the HELP screens which are of a fixed format. While Crystal permits the underlying logic to displayed as a HELP screen at any stage by pressing the F1 function key, this key has been disabled in the final version since the coded rules are generally meaningless except to the knowledge engineers. However this ability to check the logic is crucial to the knowledge engineers during program development and is a major advantage over algorithmic software systems

2.5 Testing and User Acceptance of WINDLOADER

Just as with algorithmic software, WINDLOADER must prove itself to be cost effective in the design office in order to gain widespread use. As well as having key experts on the advisory committee, five practising engineers have closely assisted in the design of the user interface and testing of the software. Four prototypes were released over the development period and tested on routine designs ranging from houses to multistorey buildings.

The code developers were greatly assisted by the development since over 20 logical inconsistencies, ambiguities and omissions were detected. All of these would probably have remained obscure until the code was released and engineers started to use it since such errors can be very difficult to spot from reading the code. Omissions were the main problem and the programming style of expert systems makes detection of these more obvious than algorithmic programming unless a truth table type approach is adopted.

In one case a contradiction was also discovered in which three successive tables gave different pressures on a flat roof of a symmetrical square building for the same wind speed from different directions. As a result, columns had to be removed from two of the tables and a new column added to the third.

A benefit of the software development is that WINDLOADER can more readily evaluate the full range of wind directions and resulting pressures on building faces whereas this would be too time consuming by hand. For example, in the case of 8 wind directions with different wind speeds for the capital cities, a building with 6 faces (4 walls and 2 pitched roof faces), may have in excess of 48 pressure combinations, all of which can be calculated by WINDLOADER. This will allow all pressure combinations with dead and live loads to be considered in order to determine the worst loads on a structure. However users following the code manually are likely to only consider a few of the pressure combinations and hence they can never be certain if the worst loading cases have been overlooked.

3. BUILDING CODE OF AUSTRALIA (BCA)

An expert system for the Building Code of Australia is being developed in collaboration with the Australian Uniform Building Regulations Coordinating Council (AUBRCC). The BCA will be simultaneously restructured to become a performance-based code and the project is expected to take about three years. A demonstration prototype for a small part of the code has been developed using the Crystal PC expert system shell and this was instrumental in AUBRCC deciding to collaborate in further development.

The BCA does not have any complex equations like WINDLOADER but this is offset by the greater size of the BCA. The BCA is also undergoing extensive logic restructuring and one of the key experts assisting with this restructuring will also be assisting with the expert system. The development of the prototype has indicated that while the BCA is rule based and thus could be

directly coded into a rule-based system with depth-first search, it is much more convenient for the user if a menu-based format is adopted to allow much faster processing and removal of unnecessary questions.

4. CONCLUSION

The project has proved that a PC-based expert system with a reasonably fast response time can be developed for a relatively complex design code and integrated with algorithmic procedures. However some of the features desired in an expert system, such as reasoning from a transparent knowledge base, have to be compromised as a result of having equations, table interpolations and large arrays of data to be stored and processed. As a result the user can only access formatted help screens but this is offset by the convenience of other features such as graphics to enhance explanations and a more user friendly interface via menus.

It would not have been possible to complete this work in reasonable time without the use of a commercial expert system shell which permitted extensions to be added in C. Other more powerful shells are becoming available and it is felt that these will make the task easier in future, especially for those shells using a common portable language such as C.

ACKNOWLEDGMENTS

The authors are grateful to Standards Australia and its subcommittee, BD/6/2, plus the Department of Industry, Technology and Commerce for their support of this project over the 1987-89 period. Speeial thanks are due to Ms F. Ho, Dr J. Holmes, Ms C. Jeammes and the SA/CSIRO WIND-LOADER advisory committee for their extensive efforts in making the project a success.

REFERENCES

1. FENVES S.J., RANKIN K. and TEJUJA H.K., The Structure of Building Specifications, NBS Building Science Series 90, National Bureau of Standards, Washington DC, 1976.

2. MARKSJÖ, B.S. and HATJIANDREOU, M., Developing Knowledge-Based Systems for Building Design Approval, Proceedings of the I.E.Aust National Engineering Conference, Melbourne, March, pp.206-210, 1985.

3. STONE D. and WILCOX D.A., Intelligent Systems for the Formulation of Building Regulations. Proceedings of the 4th Int. Symp. on Robotics and Artificial Intelligence in Building and Construction, Israel, 22-25 June 1987.

4. FENVES S.J., What is an Expert System, in Expert Systems in Civil Engineering (eds Kostem, C.N. and Maher, M.L.), American Society of Civil Engineers, New York, pp. 1-6, 1986.

5. STANDARDS AUSTRALIA, Minimum Design Loads on Structures, Part 2 - Wind Forces, AS 1170.2, 1989.

6. MELCHERS R.E., Human Error in Structural Analysis, Proceedings of Seminar on Quality Assurance, Codes, Safety and Risk, Dept of Civil Engineering, Monash University, pp.91-94, 1984.

7. LE TEXIER J.Y., DOMAN A., MARKSJÖ B.S. and SHARPE R., Use of Prolog with Graphics including CAD, Proceedings of Ausgraph 85, Third Australian Conference on Computer Graphics, Brisbane, pp.81-85, 1985.

8. THOMSON J.V., BIRD S., THOMSON D., MARKSJÖ B.S. and SHARPE R., Extending Prolog to Provide Support for Design Code Expert Systems, Microcomputers in Civil Engineering, 3, pp.93-109, 1988.

Leere Seite
Blank page
Page vide

Incorporation of Steel Design Codes into Design Automation Systems

Incorporation des normes de construction métallique dans des systèmes automatiques de projet

Einbeziehung der Normen für Stahlbau in automatische Projektsysteme

Bruno FEIJÓ
Aeronautics Engineer
PUC/RJ
Rio de Janeiro, Brazil

Patrick J. DOWLING
Professor of Civil Eng.
Imperial College
London, UK

David Lloyd SMITH
Civil Engineer
Imperial College
London, UK

Bruno Feijó received his Engineering degree at ITA, São Paulo, 1975 and his PhD at Imperial College, 1988. For many years he worked with computers in engineering design. He is now Lecturer of Computer Science in PUC/RJ. He is also Visiting Lecturer in the Expert Systems Lab, Imperial College.

Patrick Dowling is a member of Britain's National Academy of Engineering, the Fellowship of Engineering. He is Chairman of the Drafting Panel of Eurocode3 and Editor of the Journal of Construction Steel Research and Director of the Consulting firm Chapman & Dowling Associates Ltd which drafted the Brazilian Steel Bridge Code.

Dr David Smith is Head of the Systems and Mechanics Section of the Civil Engineering Department, Imperial College, London, and he is currently Director of Undergraduate Studies. He plays a leading role in the Civil Engineering Expert Systems Laboratory.

SUMMARY

This paper presents a model for the representation of design codes based on first-order logic and their incorporation into Design Automation Systems. Also, a formal link with hypertext systems is suggested. Furthermore, presented are the results of the initial investigation into the logical structure of the British Code BS5950 for steel design.

RESUME

Cet article présente un modèle pour la représentation des normes de projet basé sur une logique de premier ordre et leur introduction dans des Systèmes Automatiques de Projet. Une liaison formelle avec un système de type hypertexte est également présentée. Les résultats d'une investigation préliminaire sur la norme anglaise BS5950 concernant les structures métalliques sont également présentés.

ZUSAMMENFASSUNG

Dieser Artikel stellt ein Modell für die Beschreibung von auf logischen Grundregeln aufbauenden Projektnormen und deren Einbeziehung in automatische Projektsysteme vor. Eine formale Verbindung mit Hypertext Systemen wird auch angeregt. Die Ergebnisse einer ersten Analyse der logischen Struktur des British Code BS5950 für Stahlbauprojekte werden präsentiert.

1. INTRODUCTION

This paper presents a model for the representation of design codes that can be easily incorporated into Design Automation Systems. This model is based on and-or graphs and first-order logic. Furthermore, this model suggests a formal link between design codes and hypertext systems - an emerging field in engineering information management. Also, presented are the results of the initial investigation into the logical structure of the British Code BS5950 for steel design. The practical experience of the authors with the present model is restricted to the domain of steel design. However, the model may be applied to any design code.

2. THE PROBLEM OF KNOWLEDGE REPRESENTATION

The study of regulations as a type of knowledge can be found in the areas of Office Automation [1] and Legal Reasoning [2][3]. In Design Research, this type of knowledge presents the following characteristics:

- it is supposed to have an explicit and precise interpretation (in contrast with the "open nature" of law in the field of legal reasoning);
- it is available in written form;
- it is supposed to be complete and correct;
- it presents no uncertain facts;
- it requires no vague reasoning;
- it presents a simple structure of discourse (in contrast with the complex structures found in the area of Text Generation [4]).

The representation of this knowledge in an artificially intelligent system involves two main tasks: its structuring and the translation of the written provisions into a form suitable for symbolic manipulation. However, these tasks are not easily accomplished, because they should be carried out with the entire design process in mind. Moreover, revisions of the text are required when the knowledge is incomplete and/or incorrect[(a)]. In this case, a knowledge engineer might be required to reveal the results intended by the code writers.

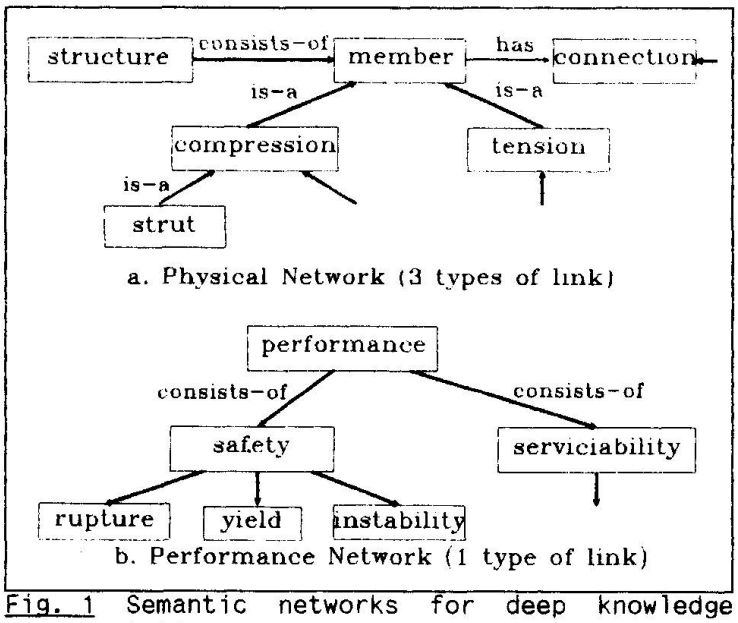


Fig. 1 Semantic networks for deep knowledge representation

Any type of knowledge representation in design should use deep models (i.e. models with a causal structure underlying the more external structure). As far as design codes are concerned, any deep model will certainly be biased by the work of Fenves, Wright and Harris [5][6]. In this

context, one could produce semantic networks like those in Fig.1. This paper assumes the existence of such deep structures and focusses on more external representations of a design code. The proposed representation is based on and-or graphs and first-order logic.

3. THE GRAPH REPRESENTATION

3.1 The design code graph

The structure of a design code can be represented by an and-or graph[b] in which provisions call subprovisions (Fig.2). Furthermore, attributes A_i are attached to each node i. These attributes contain additional information about the provisions, such as the Limit States (LS) governing the provision, the number of the section, a copy of the original text and one or more labels for classification. For example:

A_3 =
{ LS(yielding, rupture), 4.6, "For tension members with...", ... }

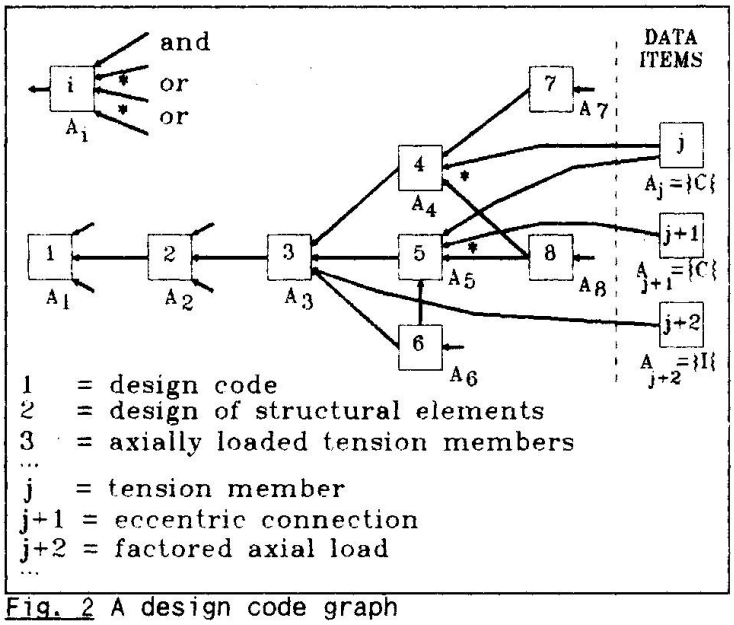


Fig. 2 A design code graph

Data items must be added to the and-or graph as terminal nodes. A data item should be identified by its type (which is an attribute) according to the following classification:

I INPUT. For example: "factored axial load";

C CLASSIFIER. It is a special kind of input that classifies an entity and makes a provision entitled for conformance checking (i.e. the process of checking a design entity for conformance with a code). For example: "tension member";

D DEFINITION. It is a data item used in definitions. For example: "the area of a hole is calculated in the place of its axis" is used in the definition of "hole area";

E EXTERNAL. It is a data item that is given after an external procedure is executed. Sometimes the external procedure is a call to the design code itself. For example: checking a double angle requires that "the slenderness of each component does not exceed 80" (in turn, this may require a consultation of the code).

The and-or graph with attached attributes and data items is called a design code graph.

3.2 Properties

Design code graphs present the following properties:

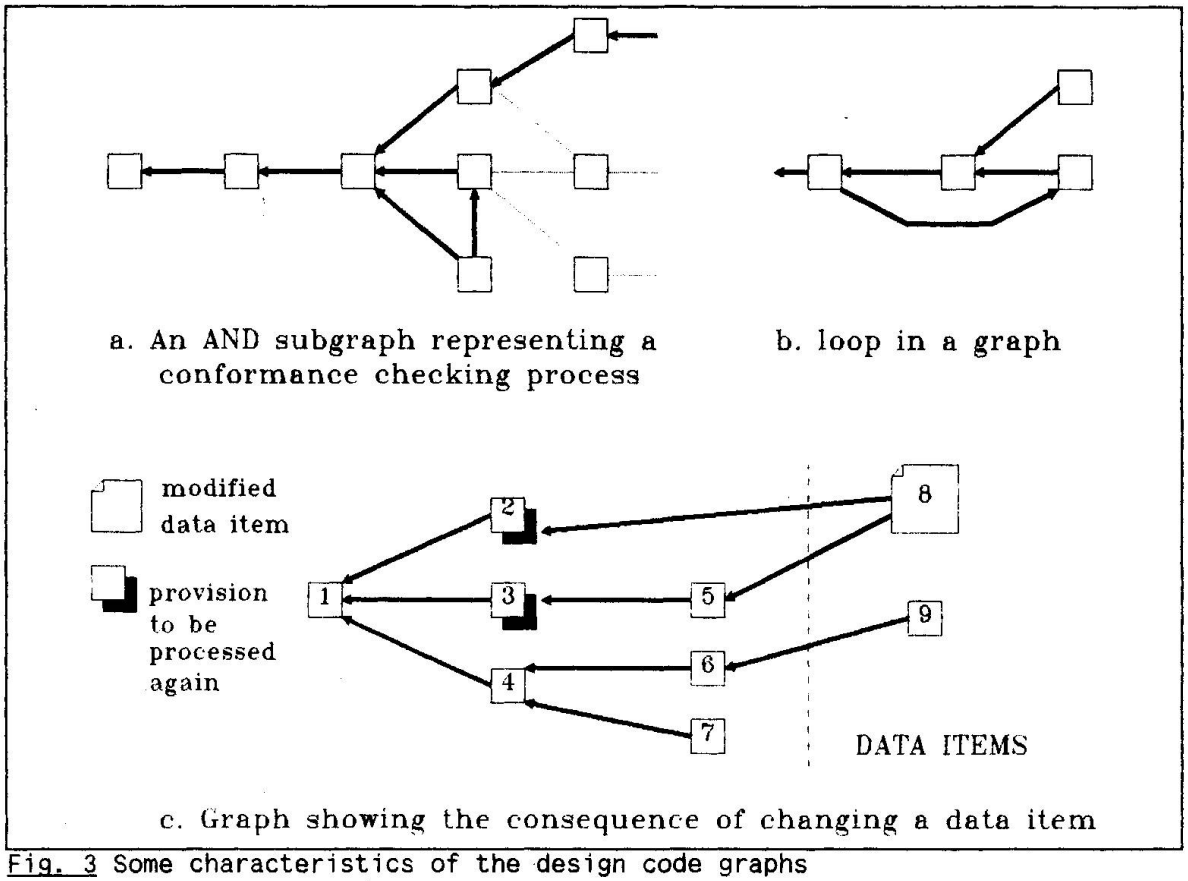


Fig. 3 Some characteristics of the design code graphs

1. Conformance checking should be represented by a unique AND subgraph, otherwise the code presents ambiguities or multiple interpretations (Fig. 3.a).

2. Cumbersome forms of cross-referencing, lack of connections and loops (Fig. 3.b) can be easily detected with the help of the design code graphs.

3. The graphs (and especially the subgraphs) show the consequences of changing a particular data item (Fig. 3.c). This property can be used to support "intelligent decisions" made by Design Automation Systems. Also, it allows an effective redesign.

4. Design code graphs share some properties with the information networks proposed by Fenves and Wright [5]. In particular, the graph can easily be converted into a depth-first spanning tree [7]. This tree can be used by writers of design codes to identify cross references and to achieve better textual expression.

4. THE HORN CLAUSE SYSTEM

The design code graph of Figure 2 does not represent a design code in its entirety, albeit it represents its overall organization. The representation is completed by attaching a set of Horn clauses H_i to each node i. These Horn clauses must be mutually exclusive rules which should define the provisions completely (Fig.4). Moreover, the set H_i can contain conditions which are not presented in the and-or graph, such as equations. The mutually exclusive rules assure the property of uniqueness (i.e. that the design code yields one and only one result).

The main variable X in the set of Horn clauses represents an entity to be designed, such as a member or an element of a member. Only one kind of entity

should be associated with the design code each time the code is invoked. This simple semantics yields a more robust model.

The union of the sets H_i forms a logic program P, called a design code program, i.e.

$$P = H_1 \cup H_2 \cup \ldots \cup H_n$$

The program P (or any subset of it) should be invoked by a query that extracts only one answer. This restriction is required by the property of uniqueness.

The most important characteristic of P is that it contains the structure of the graph that uses P to be defined. This recursive aspect of the model yields a simple and concise conclusion:

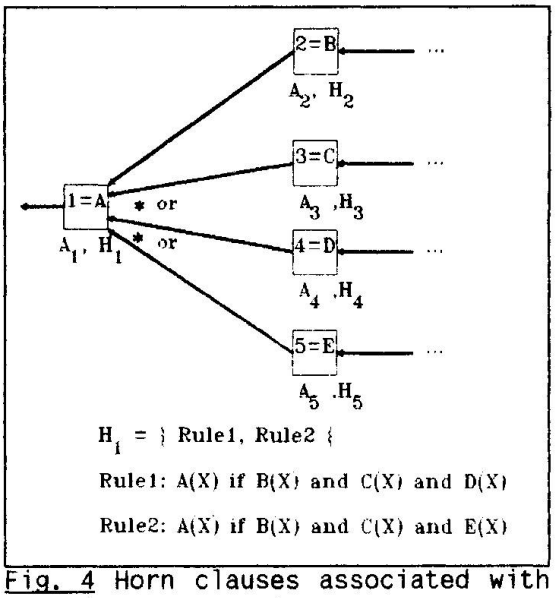


Fig. 4 Horn clauses associated with a design code graph

"a design code can be entirely represented by the program P if the attributes A_i are attached to each member of H_i".

For example, the provision A of Fig.4 might be represented by:

A(X) if B(X) and C(X) and D(X) ; A_1
A(X) if B(X) and C(X) and E(X) ; A_1

In this approach, each rule contains two kinds of information: one concerning logic (inherent to any logic program) and the other concerning the structure and organization of the code (the design code graph).

5. A RESTRICTED FORM OF HYPERTEXT

The design code program presented in Section 4 can be understood as a restricted form of hypertext[c]. In this case, nodes are predicates, links are represented by the logic program P and node bodies are the attributes A_i. Naturally, the properties of design code graphs are valid for this form of hypertext which is called a design code hypertext. A prototype of this hypertext system has not yet been implemented by the authors.

Some of the principles underlying the system KMS [8] seem to be very appropriate to design code hypertexts. Hence, nodes could be displayed as windows with four components (Fig.5): Node Header, Node Body, Logical Items and Command Items.

In the design code hypertext there is just one type of link (although a second type similar to KMS Annotation Items could be considered). Hence, a link is not an object but a property of an item. The process of editing nodes is supposed to be similar to that found in KMS.

Every aspect of the design code hypertext has a counterpart in the logic program P and vice-versa. For example, querying P is like an operation of automatic navigation. Moreover, if the querying mode is interactive (i.e. the system stops and asks for missing facts), then conventional navigation might be available temporarily. Also proof trees might be available for conventional navigation. There are many other aspects to be explored, such as: Selection by freezing

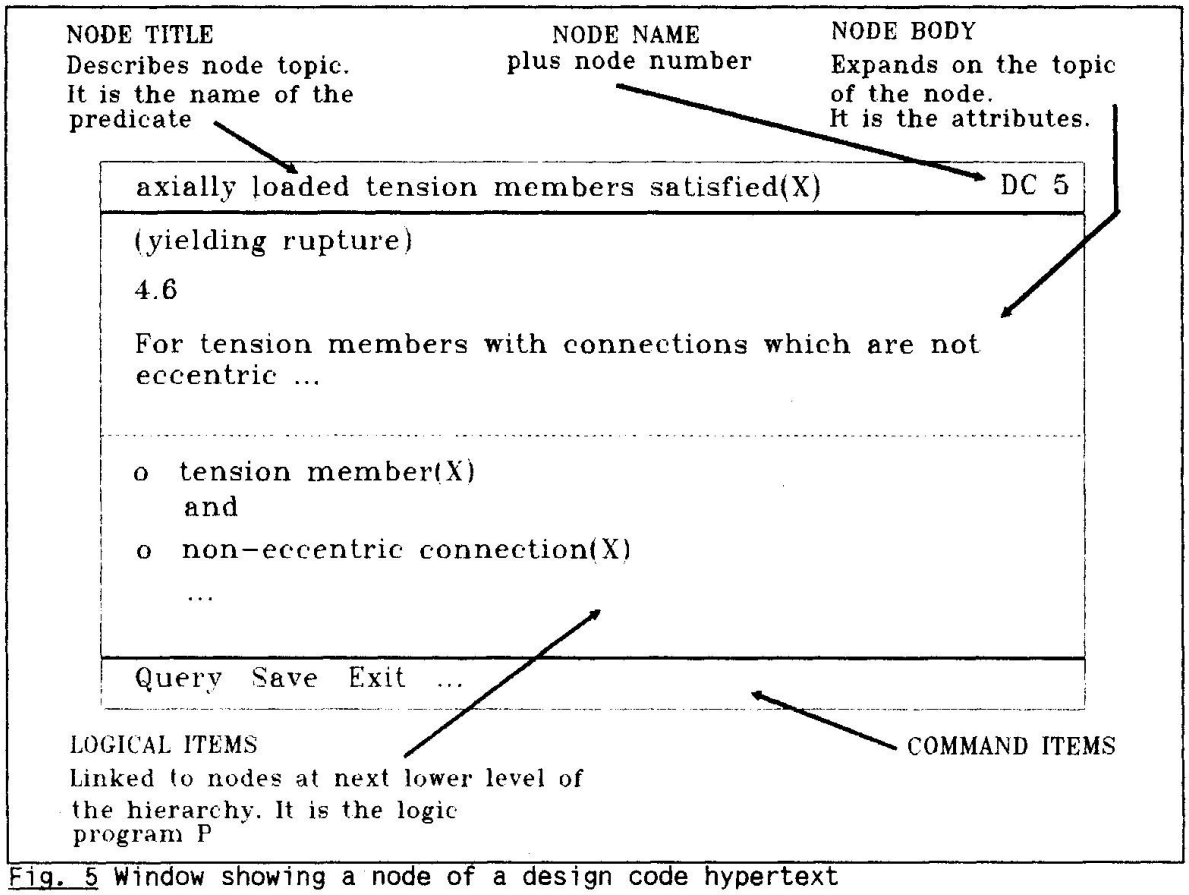


Fig. 5 Window showing a node of a design code hypertext

undesirable provisions (e.g. those concerning serviceability as limit states); Priority for searching (e.g. yielding provisions first); Consequences of changing data items; Dynamic incorporation of provisions, e.g. A(X) if B(X) and assert(A(X)); Explanation; Analogy based on past episodes; Debugging. For instance, a unique metaknowledge can be used to guide both the logical inference and the hypertext navigation (by freezing part of the program/network).

6. DESIGN CODES AND DESIGN AUTOMATION SYSTEMS

Design codes can be effectively incorporated into Design Automation Systems if their models have the same mathematical formalism. This common framework, the authors believe, should be first-order logic.

The model SAE [9] is based on first-order logic and considers design as a recursive process involving Synthesis, Analysis and Evaluation. In this model, the current state of design is represented by a set of facts in predicate form, e.g. "tension-member(b1)". These facts build up the Database of Design Facts (DDF) and are in canonical form (i.e. the standard form used by a specific design code). The translation of a fact from a general form into a canonical form should be done at a higher level process of the model SAE.

The facts in DDF are arranged in three areas that are interpreted by the design code model as follows:

- **DESIGN REQUIREMENTS.** These facts represent a design request which includes hard constraints and basic assumptions. For example: "tension-member(beam1)";

- **DESIGN OPTIONS.** These facts are soft constraints from a design script. For example: "lacing-system(m1)";

- **DATA.** These facts are temporary data retrieved from the conventional database or generated by a subprocess. For example: "area(beam1, 0.5)" from the database; "tension-capacity(beam1, 235)" deduced within a provision; "axially loaded tension member satisfied(beam1)" as a provision that was satisfied.

Synthesis processes may decide to move a fact from an area to another area at any time.

The design code model proposed in this paper can be used by the model SAE in several ways. First, a higher level process (e.g. Preliminary Design) can use the design code as a logic program for determining appropriate design requirements. These requirements (usually related to limit states) are then used as hard constraints (i.e. performance specifications). For example: maximum thickness, load factors and maximum deflection.

Second, conformance checking can be invoked at any time as an analysis process, what is usually done in a lower level (e.g. Detailing). In this case, the query to the logic program can invoke the entire code, e.g.

BS5950_satisfied(beam1) ?

or part of it, e.g.

axially_loaded_tension_member_satisfied(beam1) ?

If a query fails, the reason for failure (a fact) should be available for some sort of intelligente decision making.

Third, the logic program can use a built-in predicate (e.g. assert) to add provisions to DDF if they hold, e.g. A(X) if B(X) and assert(A(X)). This technique saves processing time if those provisions are called again.

Forth, meta-knowledge can be exercised in the present model by using the attributes A_i. Finally, tests of inconsistency and redundancy during the process of using the design code (see Chapter 7) can be easily implemented. A preliminary version of a design code based on the present model was written in C by the authors.

7. INCONSISTENCY AND REDUNDANCY

The user or any Artificial Design Assistant of the model SAE may introduce inconsistencies into the knowledge database at any time. For example, he/she/it may state that the member is a "tension member" and later state that the member is a "strut" (i.e. a member of a structure carrying predominantly compressive axial load), which is a contradiction. There is also the problem of redundancy. For example, he/she/it may state that "the grade of steel is 43A" and "the flange thickness is 16 mm", and may state later that "design strength is 275 N/mm^2". This last sentence is a redundancy[d], because design strength is a logical consequence of the previous data.

First-order logic allows one to look for classical inconsistencies introduced into the database each time a new fact is given. For example, if "strut(beam1)" is given to the following knowledge database:

1. if strud(X) then compression_member(X)

2. not(tension_member(X) and compression_member(X))
3. tension_member(beam1)

it will create an inconsistent knowledge because both "strut(beam1)" and "not strut(beam1)" are true.

The procedure for redundancy tests is the following: a new fact should not be added to the database if it could be obtained as a consequence of the logical system. In very large and complex knowledge databases, it should be more practical to reject facts that can be proved in less than a specific number of steps (say 3 or 4).

8. THE DESIGN CODE BS5950

An investigation into the structure of the code BS5950 [10] has revealed some problems. First, the design code graph of the current version of the code presents a cumbersome structure due to an excess of cross referencing, as shown in Fig.6.

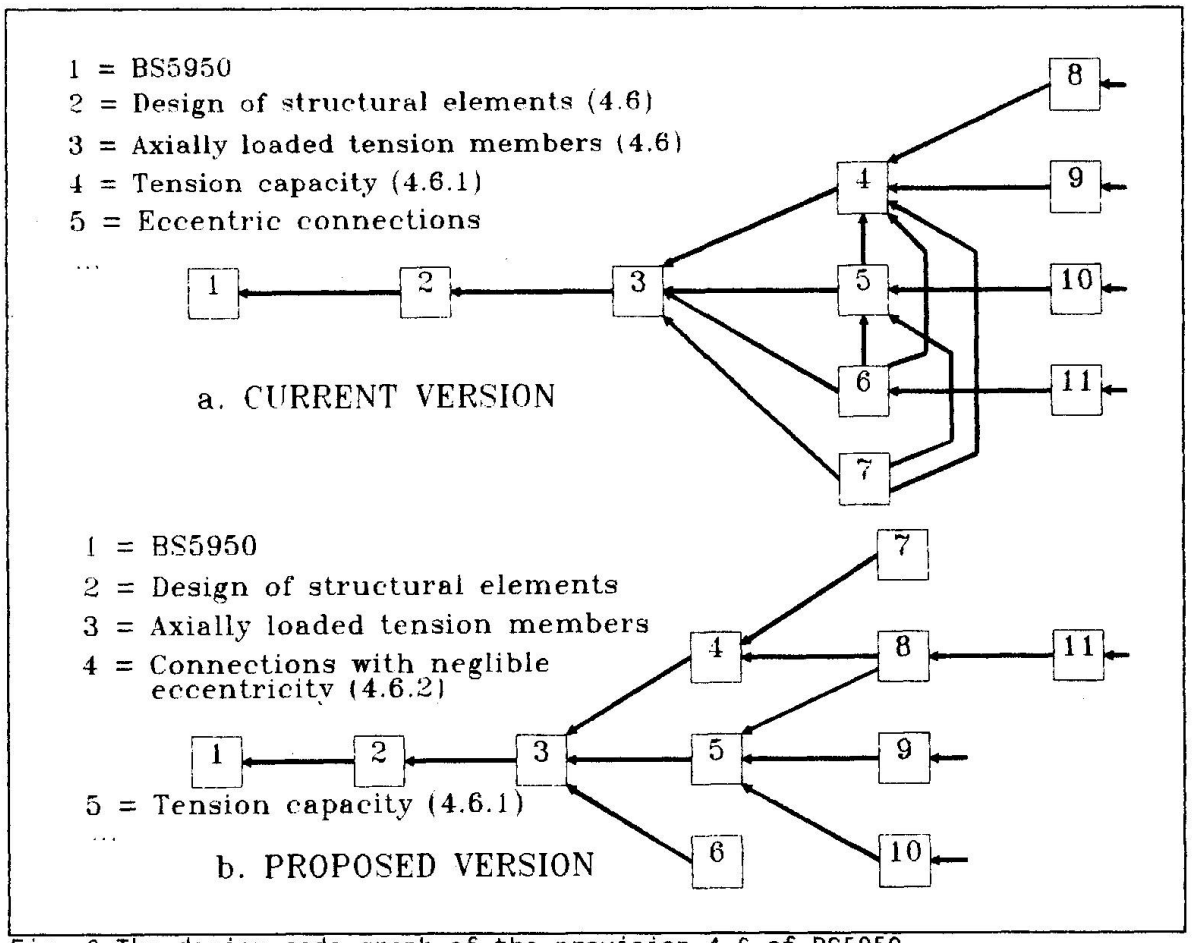


Fig. 6 The design code graph of the provision 4.6 of BS5950

Second, some provisions provide no general guidance at the first entry level of the provision (e.g. provision 4.6).

Third, the knowledge embedded in some provisions is not clearly expressed in the text. For example, the actions to be taken in provision 4.6 are not explicitly related to the three exclusive conditions about connections: (1) not eccentric; (2) eccentric; (3) eccentric, but the effect of the moments may be negleted.

Forth, the original text contains the objectionable word "except". This word has a metalevel interpretation in logic ("expect A" means "if A cannot be proved") that complicates deductions for humans. The semantics of "except" is a special case of negation called "negation as failure". Although this type of negation solves many problems it is advisable to avoid it (specially if nested negations may occur). This word could be easily removed from a design code by changing the structure of some provisions.

Fifth, some provisions permit undesirable logical conclusions. For example, from the point of view of methematical logic, the section 3.4.3 allows one to design any configuration of staggered holes. However, this is not the intention of the code writers. Irregular lay-outs of holes should not be permitted because the provision made in 3.4.3 is based upon empirical conclusions for regular distributions of holes.

9. CONCLUSIONS

This paper shows that logic leads to a simple, compact and robust representational model of a design code. Furthermore, logic can be used as a common framework for the design code model and the design process model. In this approach, inconsistency is gracefully handled by using classical negation in logic. Moreover, redundancy tests are easily implemented by a deduction mechanism.

The proposed design code model is simpler and, in some aspects, more robust than those proposed by other authors [11][12][13]. However, many aspects of those models can coexist with the present model. For instance, decision tables can be used by the knowledge engineer as a support tool during the building of a design code program.

Some problems of specific design codes may be revealed during the building of a design code program. For example, a partial analysis of BS5950 revealed cumbersome forms of cross referencing, hidden knowledge and texts that permit undesirable logical conclusions. These distortions in the structure of a code cause no harm to experienced engineers. However, Design Automation Systems require a clear and formal representation of the design code.

There are many ways in which the present work might be extended, such as: incorporating the design code model into systems with deep knowledge; further investigation into the hypertext nature of the model; a complete analysis of a code and its issue as a logic program or a hypertext system (as an alternative to its textual version); the study of problems associated with very large knowledge databases; further analysis of design codes in the light of the methods found in the field of Text Generation and Understanding.

NOTES

a) Incorrect in the sense that the results are not those intended by the code writers.

b) By definition, and-or graphs assume the inclusive interpretation of "OR", i.e. at least one (but possibly more) subprovisions hold.

c) Hypertext (or generally speaking: hypermedia) is a form of eletronic document in which data is stored in a network of nodes connect by links. Nodes can contain text, graphics, sound, programs to be executed or other forms of data. The entire network and individual nodes are displayed through an window system.

Users navigate in a hypertext database by pointing the mouse cursor at an item which has a mark to indicate a link to another node.

d) It could be an inconsistency if a contradictory value is given, e.g. "design strength is 355 N/mm^2".

ACKNOWLEDGEMENTS

The authors would like to thank the Steel Construction Institute for the financial assistance.

REFERENCES

1. PAOLINI,P. et al., Knowledge based document generation. In Lamersdorf, W.(ed.), Office Knowledge: Representation, Management, and Utilization, Elsevier Science Publ., North-Holland, IFIP, 1988, p.179-195.

2. SERGOT,M., KOWALSKI,R.A., KRIWACZEK,P. HAMMOND,P. and CORY,H.T., The British Nationality Act as a Logic Program, Commun. ACM, 29, (5), May 1986.

3. PINHEIRO,C.S. and SCHWABE,D., Expert systems and social welfare benefits regulations: the Brazilian Case, Comp. Science Dept, PUCRio, Brazil, 1988.

4. MANN,W.C., Text generation: the problem of text structure, in McDonald,D. and Bolc,L. (eds.), Natural Language Generation Systems, Springer-Verlag, 1988, p. 47-68.

5. FENVES,S.J. and WRIGHT,R.N., The representation and use of design specifications, in Hall,W.J. (ed.), Structural and Geotechnical Mechanics, Prentice-Hall, 1977, p. 278-304.

6. HARRIS,J.R. and WRIGHT,R.N., Organization of building standards: systematic techniques for scope and arrangement, NBS Building Science Series 136, 1981.

7. AHO,A.V., HOPCROFT,J.E. and ULLMAN,J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

8. AKSCYN,R.M. et al., KMS: a distributed hypermedia system for managing knowledge in organizations, Commun. ACM, 31, (7), July 1988, p.820-835.

9. BENTO,J, FEIJO,B. and DOWLING,P.J., The knowledge based design of steel portal frames for agricultural buildings, IABSE Colloquium, Expert Systems in Civil Engineering (to appear).

10. BRITISH STANDARDS INSTITUTION, BS5950: Structural Use of Steelwork in Buildings, Part 1, England, 1985.

12. ROSENMAN,M.A., GERO,J.S. and OXMAN,R., An expert system for design codes and design rules, in Sriram,D. and Adey,R. (eds.), Applications of Artificial Intelligence to Engineering Problems, Springer-Verlag, 1986, p.745-758.

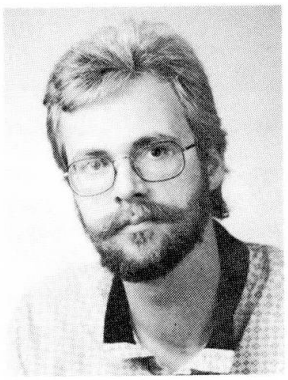
13. THOMSON,J. et al., Extending Prolog to provide better support for design code expert systems, Microcomputers in Civil Engineering, 3, 1988, p.93-109.

Integrated Expert System for the Design of Steel Structures

Système expert intégré pour l'analyse des constructions en acier

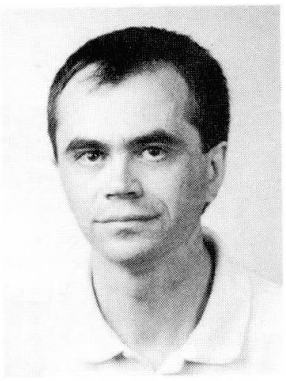
Ein integriertes Expertensystem für normgerechte Nachweise im Stahlbau

Christof H. SCHÜRMANN
Dipl.-Ing.
Ruhr-University
Bochum, Fed. Rep. of Germany



C.H. Schürmann, born 1960, received his degree in civil engineering at the Ruhr-University of Bochum, FRG, in 1986. Since then he has been active in research on integrated, microcomputer-based design strategies in structural engineering on microcomputers. His special research field are knowledge-based approaches of design strategies.

Burkhard WEBER,
Priv.-Doz. Dr.-Ing.
Ruhr-University
Bochum, Fed. Rep. of Germany



Burkhard Weber, born 1950, received his degree in civil engineering in 1974 at the Ruhr-University of Bochum. He obtained his Ph.D. in 1980 and his «venia legendi» in 1988 at the same university. Since 1981 he is head of a research group on software engineering. His main research interests are integrated software systems in CAE and CAD, using database techniques.

SUMMARY

The automation of structural analysis is accompanied by the development of up-to-date checking procedures and new ways of design. The rapid evolution of microcomputer technology has made decentral automation solutions for design tasks possible. This paper deals with a number of strategies for utilizing expert system techniques in the context of checking for accordance with code regulations for steel structures. Integrating this new technique in the existing software environment is achieved through the use of relational databases as global information support systems.

RESUME

L'automation des calculs statiques n'est plus un processus isolé; elle est suivie des concepts nouveaux de projet et de contrôle des constructions. C'est l'avancement rapide de la technologie qui a rendu possible cette automation decentralisée du processus de construction. Cette contribution présénte des stratégies pour l'utilisation des systèmes experts dans l'analyse des constructions en acier. L'intégration de cette nouvelle technologie dans l'environ de la programmation traditionnelle est rendue possible grâce aux banques des données relationales utilisées comme support de l'information.

ZUSAMMENFASSUNG

Die Automatisierung der Baustatik erfolgt nicht mehr isoliert, sondern im Verbund mit technischen Nachweiskonzepten und Konstruktionsprozessen. Die dezentrale Automatisierung des Entwurfsprozesses wird ermöglicht durch die rasante Entwicklung in der Mikrocomputerelektronik. Der vorliegende Beitrag erläutert einige Strategien zur Nutzung von Expertensystemtechniken im Bereich von Bemessungsnachweisen am Beispiel des Stahlbaus. Die Integration dieser neuen Technik in die vorhandene Softwareumgebung gelingt durch relationale Datenbanken, die als globale Informationsträger eingesetzt werden.

1. INTRODUCTION

In static analysis, strategies of automation have experienced a qualitative change due to the rapid evolution in microcomputers. The means of man/machine communication have also undergone significant changes in the course of the decentralisation of design work on microcomputers. This automation process for conducting integrated structural analysis is utilizing the latest development in software technology and focuses attention on the data links between structural analysis, dimensioning and final design. Relational databases that permit storage and easy retrieval of construction data have become key components for the information transfer between single tasks.

An important goal of integrated design is the inclusion of expert system techniques, with whose help design rules from codes, engineering experience and other sources can be readily assembled and put to use.

In this paper, first the implementation of design and checking procedures are discussed. Strategies of an expert system for checking steel buildings for accordance with the torsional-flexural buckling provisions of DIN 18800 are next dealt with. Different strategies for knowledge processing are shown and point out in which way such an expert system can effectively help the engineer carry out checking tests. The implementation of knowledge-based techniques in the existing software environment is carried out with a relational database, which communicates with the expert system via an "intelligent interface".

2. INTEGRATION OF CHECKING AND DESIGN

The integration of analysis, pre-dimensioning, design and checking is an active research field in structural informatics. Therein, the single tasks must be assembled to form a productive unit. For such an integration the modularisation of the software model and an open data transfer between the single software components with the aid of global, relational database concepts are necessary.

Through modularity the design process is split into a series of software components dealing with data input, static and dynamic analysis, dimensioning and CAD-based data output. Each software component deals with a single step of the whole design process. As an example for such a modular concept the microcomputer based program system SSt-micro (Fig. 1) [6], which permits integrated design of steel and R/C structures consisting of beam elements is presented. Static analysis therein, by plane as well as by 3D beam elements, can be done according to 1st an 2nd order theory and also based on limit state calculation. The different moduli offer the possibility to use the direct stiffness method in combination with the transfer matrix method to meet best the engineer's requirements in the refinement of results. Also moduli for dynamic analysis of structures are integrated in SSt-micro. The implemented design methods follow the German design codes DIN 18800 (steel) and DIN 1045 (concrete structures). For visualizing the data of every design step a multi-coloured grafical output is easy of access. .

The moduli are implemented in different programming languages, e.g. FORTRAN, C, LISP and PROLOG. In consequence this causes a heterogenous program- and data structure. For an integrated design work the different moduli can be combined by means of a global database, which defines all properties of a structure. The definiton is based on the different levels of design as for instance the data of static analysis, dimensioning and construction. It is open for equal data access of each modul.

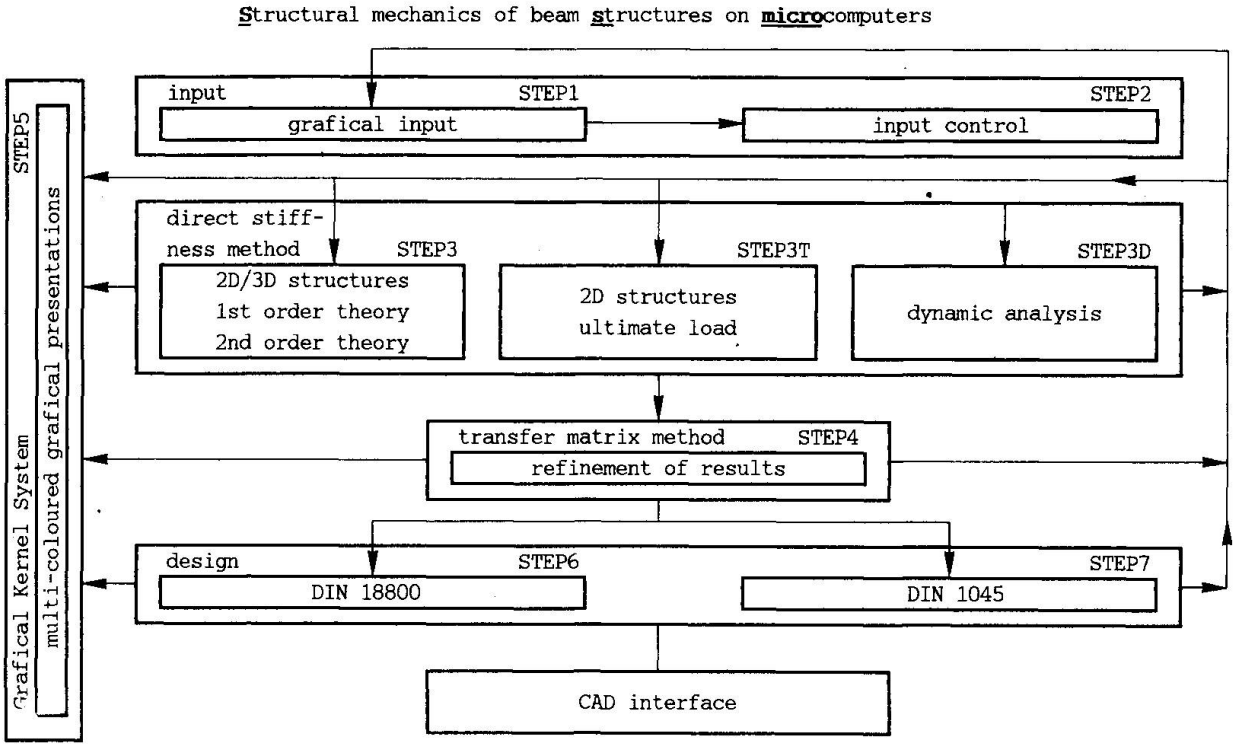


Fig 1: Modular structure of the program system SSt-micro [6]

3. KNOWLEDGE-BASED STRATEGIES FOR APPLICATION IN DESIGN OF STRUCTURES ACCORDING TO DIN 18800

Expert systems (XPS) are at the beginning of their development in the field of engineering. Some experience was available from other research fields, e.g. medical diagnosis or chemical analysis. But there are still many unanswered questions for the development and application of XPS in civil engineering problems. As a consequence of the rapid evolution in microelectronics also many powerful programming tools can be used to support the research on knowledge-based systems.

The main feature of XPS is the separation of knowledge representation and problem solving. Thus the knowledge becomes explicit for transparent representation and easy modification. The high interactivity of XPS supports the man/machine communication. The problem solving implemented on the level of meta knowledge grants an universal formulation of special engineering oriented problem solving strategies. However, the development of XPS for a special field supposes the assignment of multiple adequate knowledge representation and inference strategies. According to the German DIN 18800 some of these problems are examined.

To avoid torsional-flexural buckling the German design code DIN 18800, part two provides design rules for use in practice. This buckling problem is examined on a beam that is extracted from the complete structure [5]. The design rules are reduced to the determination of the collapse loads of an extracted beam. The torsional-flexural buckling is influenced by several factors, e.g. imperfections, section type, type and position of load and so on. The original design rules are roughly as follows:

- It is not necessary to consider torsional-flexural buckling as long as rotations of the beam end sections are adequately restrained.
- Simplified check for tors.-flex. buckl.: In all other cases (beams with arbitrary but unmovable end supports, constant section and constant axial load) pragraph 306 can be applied.
- The saftety check must be conducted in accordance with formula (301).

The whole information defined in DIN 18800 and their mutual dependencies direct the proceeding for the application of the right set of design rules. The efficient and complete definition of all relevant information and the selection of the appropriate rules for a special safety check is a main problem in using technical codes. The several design rules include different types of knowledge that can be separated into: properties of structural elements, tables, numerical functions and logical relations. While the representation of numerical functions and tables in programs is well understood, we focus on the other points mentioned above. Furthermore, the usage of a design code, e.g. the DIN 18800 can be subdivided into following steps:

- selection of a beam for the safety check,
- determination of the relavant rules and properties,
- calculation of the necessary data (e.g. the internal forces),
- safety check in accordance with the selected design rules,
- in case of rule violations: modification of properties and new safety check.

By use of knowledge processing strategies a design code can be represented in a knowledge base. The inference mechanism simulates the decision process for the safety check of the beam. A natural way formulating design codes are production rules [1]. The conditions of these rules that are proved by the inference mechanism consist of logical assertations and the conclusions are actions that take place if the conditions are true. Some design rules of DIN 18800, part two, formulated as production rules are as follows:

check of safety is par_322_304 if load_comb of beam_forces is axial_load_only and rotation of beam is restrained.

check of safety is par_322_305 if load_comb of beam_forces is axial_load_only and simpl_check of beam is true and design_rule_301 of beam is ok.

rotation of beam is restrained if symmetry of beam_sec is two_ax_symmetric and shape of beam_sec is i_shape and design_rule_305 of beam_safety is ok.

In the case of DIN 18800 and also for other codes, the goal-driven backward chaining strategy can be used to get a solution of the dimensioning problem. This strategy permits to hypothesize a potential solution, e.g.: 'The safety check of beam no. X is ok.', which subsequently can be proved or disproved. During this process all relevant rules defined in the knowledge base are automatically evaluated.

For the implementation of object-oriented knowledge processing strategies the conditions and conclusions of the rules are structured as object-attribute-value triples (O-A-V triples) to represent not only the logical relations of a special field, but also the structure of the underlying objects. The advantage of using O-A-V triples is the possibility to summarize different triples in a frame (Fig. 2) that defines a class of objects and consists of slots that hold the associated information belonging to the objects. The slots in a frame of an object can represent the possible values of the attributes or defines links to subobjects represented by other frames. Interlinking frames build a taxonomy that explicitly define the mutual dependencies of the objects of a special knowledge field. The inference strategies used in a taxonomy are manifold and

are not as strictly defined as the rule-oriented inference strategies. In a taxonomy one can use inheritance strategies, procedural attachments or define default-values [2]. Fig. 2 shows frames defining some objects of a beam structure using inheritance strategies to share common data.

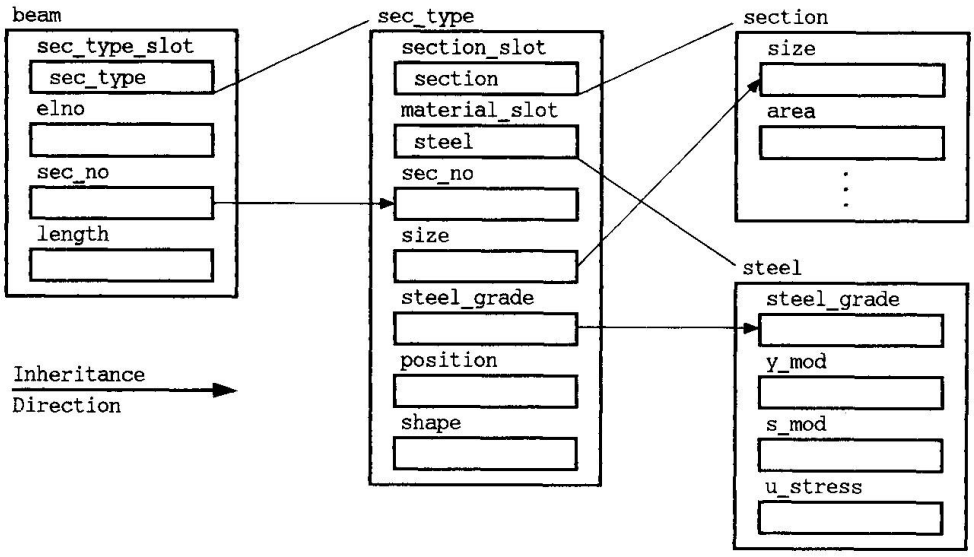


Fig 2: Object-oriented representation of structures by frames (partial view)

By the combination rule-oriented and object-oriented knowledge processing strategies the syntactical as well as the semantical aspects of knowledge can be represented which leads to the development of hybrid expert systems.

The developed expert system prototype introduced in the following chapters is implemented in PROLOG. Designing the expert system we integrated special software components delivered with the PROLOG-system for defining a rule base and a taxonomy in a pre-defined meta language [4]. The numerical procedures are implemented by interfaces to FORTRAN and C.

4. APPLICATION OF AN EXPERT SYSTEM FOR CODE-CHECKING

The following example illustrates the consultation of an expert system for the above mentioned DIN 18800. The example deals with the checking of a beam element of a frame structure. In the interactive dialog the user has to give information on the special problem while the inference process is carried out by the system. By selecting the how- or why-option the system can explain the used concepts or the drawn conclusions:

```
1) Enter steel grade of section ?

        1) st37
        2) st52
        3) unknown
        4) how
        5) why

Choice: 1

2) Enter shape of section ? heb500
3) Enter value for radius of gyration izg [cm] of section : how
```

```
Comment on design rule (308):
To simplify matters it is allowed to compute with the radius of gyration
iz instead of izg (radius of gyration of flange).
Hit any key to continue...
```

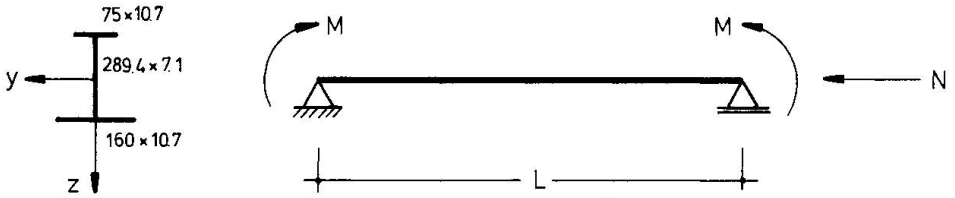
```
3) Enter value for radius of gyration izg [cm] of section : 7.84
4) Enter type of stiffening :
          1) brickwork
          2) trapezoidal_plate
          3) contin_rot_spring_supp
          4) point_supp
          5) none
          6) unknown
          7) how
          8) why

Choice : 4
           .
           .
7) Enter value of bending moment My [kNm] ? 465.84
           .
           .
The determined paragraph to avoid torsional-flexural buckling is:
          1) par_333_310_e
(e)xplain ,(c)ontinue <c> : e
```

```
[ How was check of safety = par_333_310_e determined ? ]

Since [1]  load_comb of forces = simple_bend
and   [2]  type of stiffening = point_supp
and   [3]  axial_force of forces = none
and   [4]  status_307 of design_rule_307 = not_ok
and   [5]  sym_of_web of section = yes
and   [6]  status_309 of design_rule_309 = ok
then check of safety = par_333_310_e
```

```
[ How was status_307 of design_rule_307 = not_ok determined ? ]

Since [1]  slen_rat_zg of beam = 0.6777625
and   [2]  max_value_307 of design_rule_307 = 0.5
and        0.6777625 > 0.5
then status_307 of design_rule_307 = not_ok
```

An important demand for the design of knowledge-based systems is their ability to estimate the inferred conclusions. Many complex design problems demand the use of adoptions and simplifications, which in further steps are improved to get the final results. This strategy of "iterative design processing" is a basic solution strategy in structural design.

In the case of DIN 18800 it is possible to extend the expert system to automatical search for the valid properties of a beam. All relevant rules to avoid torsional-flexural buckling are activated to check the ultimate plastic stress of the decisive section. To ensure the monotony of the used inference strategy it is necessary to eliminate all inferred conclusions in the dynamic database before execution of the next iteration step. Through the implementation of



1. design example

L = 2.50 m , max N = ?
system answer: N = 550.0 kN

2. design example

N = 550 kN , max L = ?
system answer: L = 1.61 m

3. design example

N = 200 kN , l = 2.50 m , max M_y = ?
system answer: M_y = 29.79 kN

St 37, welded section

A	=	44.62	cm^2
Z_M	=	8.65	cm
I_y	=	6172.74	cm^4
I_z	=	338.55	cm^4
I_T	=	12.64	cm^4
i_z	=	2.754	cm
i_p	=	12.08	cm
I_ω	=	28004.7	cm^6
M_{ply}	=	113.84	kNm

Fig. 3: Beam for knowledge-based analysis in accordance with DIN 18800

these iteration procedures it is possible to simulate the dimensioning process of the design examples shown in Fig. 3 by the expert system. The initial information for the iteration is incorporated into the system through a interactive dialog with the system.

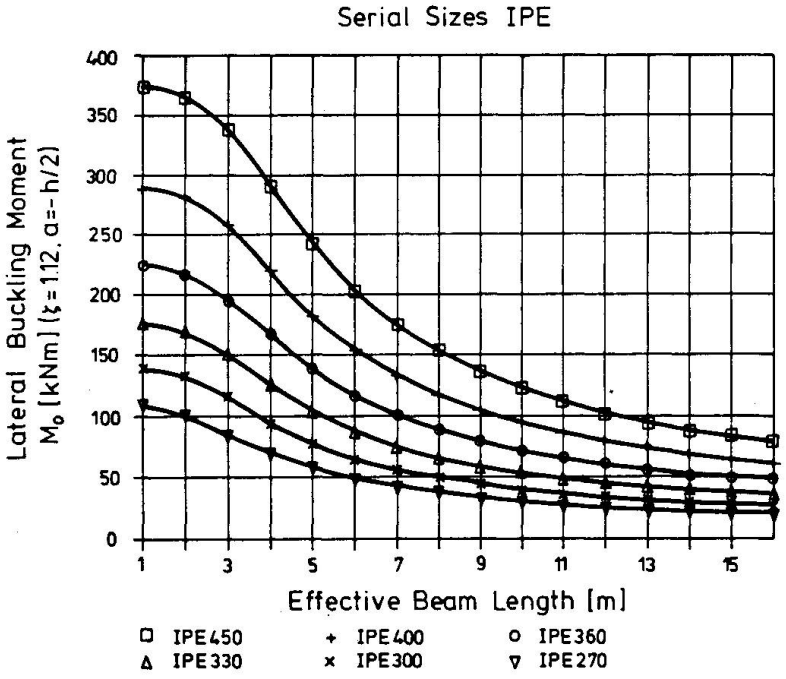


Fig. 4: Automatic generated design diagram in accordance with DIN 18800

Another utilization involving iterative processes applied to knowledge bases is the automatic generation of user defined design diagrams for special structural elements. Discrete properties of structural elements are varied automatically. The ultimate load of a beam, e.g. is determined in accordance with the code rules defined in the knowledge base. By the use of a grafical interface the resulting curves can be displayed or plotted as for instance shown in Fig. 4 for the diagram that shows design curves for the lateral buckling moment of an I-shaped beam.

5. COUPLING RELATIONAL AND OBJECT-ORIENTED DATA STRUCTURES

The duality between relational and object-oriented data structures (Fig. 5) makes it feasible to integrate knowledge-based techniques by use of a global, relational database (RDB). RDB's presently show their conductivity in the field of structural design [3]. Relational data structures transformed in 3rd normalform represent object classes equivalent to objects defined in a taxonomy. The mutual dependencies of different objects involved in the design process are defined in the relational data model and are carried out by the use of primary and foreign key attributes. A more effective and explicit definition of object structures is possible through a taxonomy using inheritance strategies. The adoption of meta relations makes it possible to implement an intelligent object manager that is under control of the expert system. The coupling between the relational data model and the taxonomy is carried out through the use of the inheritance of foreign keys from objects to subobjects of the taxonomy. The external database interface is automatically activated by way of procedural if-needed attachments when object data is needed in the inference process.

However, for an effective implementation of the model it is necessary to install a data buffer to avoid the activation of the database interface for each O-A-V triple. To minimize the external data access to the relational database the object manager only activates the interface to transfer the data of a whole object that is uniquely defined by its key attributes residing in the meta relation.

The relational data query is implemented in a special interface to the above mentioned SSt-micro system, based on SQL. With this special implementation we have the possibilty to make a fast, automatically managed, indexsequential data access to the SSt-micro database. With this model all information produced by other software components for instance the internal forces, section properties, material data, system geometry and so on, can be incorporated interactively into the expert system without any user-intervention. As a result the system dialog can now focus on the relevant information dealing with the code regulations. In future the model shall be expanded for use in a blackboard model to integrate different expert systems dealing each with separate problems of structural design.

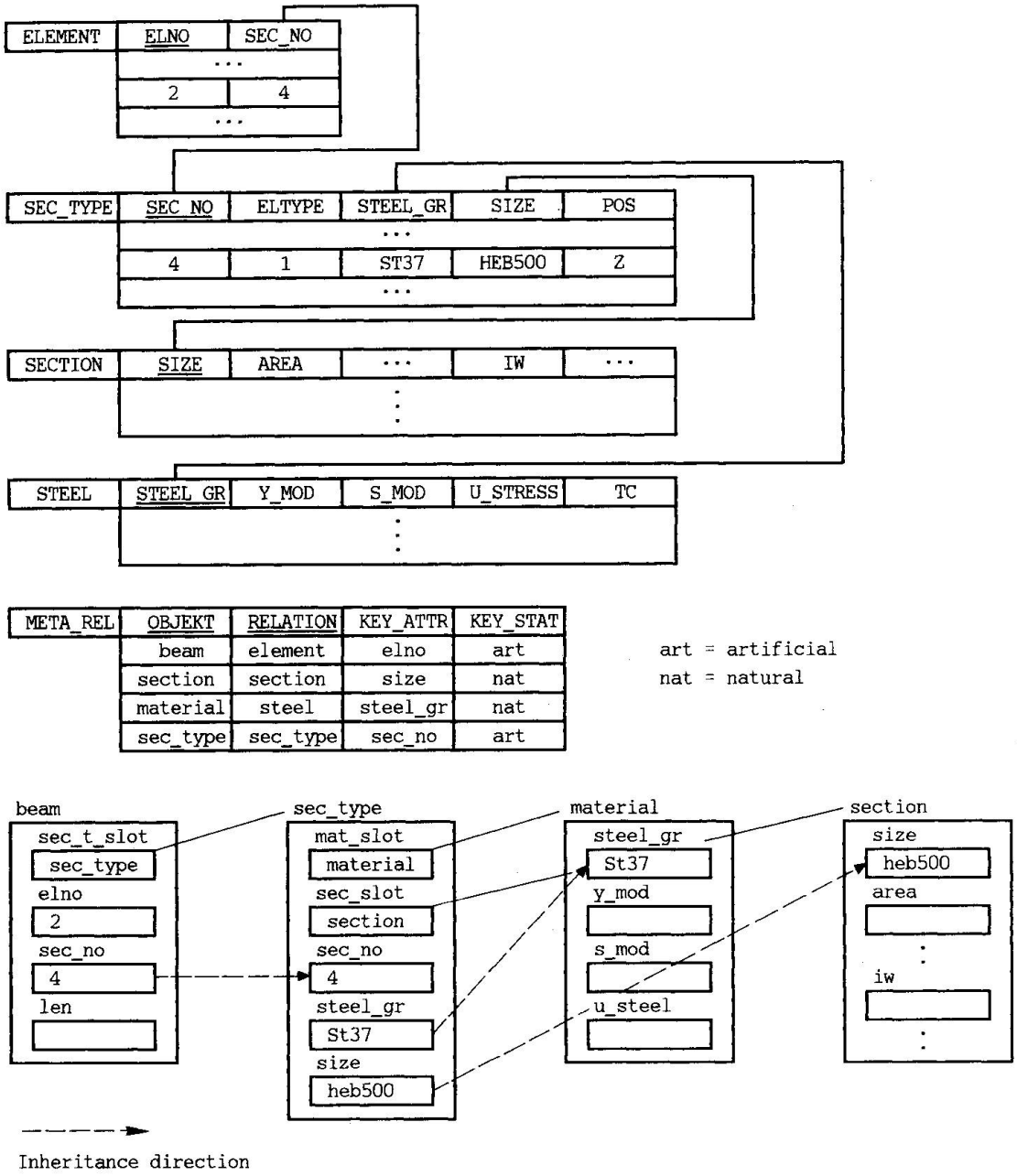


Fig 5: Duality of relational and object-oriented data structures

6. CONCLUSIONS

Codes and regulations which are a basic framework for the structural design process can be mapped and operationalized with the help of knowledge-based expert system techniques. In this way the non-numerical engineering decision process is simulated in the computer.

It has been shown that engineering problems cannot be solved all embracingly with a single strategy of knowledge processing, but the use of a variety approach can lead to successful results. The designed prototype "dimensioning expert" has a hybrid architecture to cover all aspects of the code regulation knowledge using rule-oriented as well as object-oriented techniques.

Special inference strategies have been mentioned to solve dimensioning problems of the design process as for instance the iterative checking of the implemented rules of DIN 18800 to change the properties of structural elements in accordance with the building regulations or the automatic generation of design diagrams. This leads to the development of special purpose expert systems dealing with design problems that require special expertise. The developed prototype shows the flexibility of this approach and demonstrates the great position expert systems can fill in the field of building codes and regulations.

Considering the design problem as a whole the coupling of the existing software environment and the new technology is a necessity. To cope with this problem relational database technology is used by way of a global database to coordinate the single tasks of the design process. That has been shown by the developed model for coupling relational and object-oriented data structures.

The hardware requirements of expert systems can be realized by today's 32-bit microcomputers, but they overcharge the current standard configuration of microcomputers used on top of the engineers desk. However, in a few years the application of expert systems in structural design will be widespread. In consequence the way of dealing with the computer and the design process itself will undergo a qualitative change. The final result of this evolution must not replace the human engineer but instead he should be extensively supported by expert systems to get more opportunity for creative engineering performance.

ACKNOWLEDGEMENTS

The presented paper was prepared at the Institute for Statics and Dynamics (o. Prof. Dr.-Ing. W.B. Krätzig) of the Ruhr-University of Bochum. The authors express their gratitude to the Minister for Science and Research of NRW (FRG) for his generous financial support.

REFERENCES

1. ADELI, H., Expert Systems in Construction and Structural Engineering. Chapman and Hall, 1988.

2. COYNE, R., Logic Models of Design. Pitman Publishing, 1988.

3. KRÄTZIG, W.B., SCHÜRMANN, C.H., WEBER, B., Integrierter Ingenieurentwurf auf Mikrocomputern. In: Wunderlich, W., Stein, E. (Ed.), Finite Elemente - Anwendungen in der Baupraxis, Ernst & Sohn Berlin, 1988, pp. 441-452.

4. N.N., The Arity/Expert Development Package. Arity Corporation, 1986.

5. ROIK, K., KINDMANN, R., Das Ersatzstabverfahren - Eine Nachweisform bei planmäßig einachsiger Biegung mit Druckkraft. Der Stahlbau, 12/1981, pp. 353-358.

6. WEBER, B., DUBOIS, I., SCHÜRMANN, C.H., SSt-micro: Statik der Tragwerke auf Mikrocomputern. Institut für Statik und Dynamik, Ruhr-Univ. Bochum, 1988.