Blackboard Based Expert System For the Reinforcement Layout

CHRISTOPH MEINECKE
Dipl-Ing, Research Assistant

RAIMAR J SCHERER
Prof Dr-Ing, Professor
Institut fuer Massivbau und Baustofftechnologie
(Institute of Structural Concrete and Building
Material Technology)
University Karlsruhe
D - 7500 Karlsruhe 1
Germany

ABSTRACT

This paper presents a design expert system for the layout of the reinforcement for concrete members. The system has a blackboard architecture and a hybrid structure to integrate different sources. A domain independent process model is implemented to represent the design cycle. The domain knowledge is encoded in rules and in object-oriented structures. Hierarchical planning is used in order to decompose the design task to gain more clarity for the design elements.

Key Words
reinforcement design; design expert system; blackboard architecture; design cycle; hierarchical planning

Introduction

Today, the drawing and the presentation of the designed reinforcement for concrete members is often done with a CAD system. The design process itself, however, is still handmade. In order to develop a knowledge-based system which accomplishes the design task it is recommended to decompose the task into smaller functional elements (Scherer, 1990 b). These elements have certain inner relations and interdependencies, but they can be nevertheless elaborated separately. The design strategy of hierarchical planning provides a decomposition of the design task in subsequent levels where a higher level depicts an abstraction of the following lower levels.

The hybrid expert system presented here combines a top-down with a bottom-up modelling technology. An engineering system can be decomposed very deeply into technical or physical levels. Such depth of levels of abstraction is not worth striving for because the advantage of structuring will be lost by the complex management of the levels of abstraction. Therefore, the top-down technique is used until an abstraction level is reached where solutions for the local design task can be easily formulated in a bottom-up manner (Scherer, 1993).

Architecture of the System: Blackboard Structure

As a whole the system consists of the following modules (fig. 1):

- An object-oriented data base which is substructured in knowledge bases or taxonomies.
- A context tree with its alternative knowledge bases, the first represents the already achieved design the second embody the alternatives for the actual design focus.
- A control mechanism and a process model to manage the evolution of the context tree.
- A rule base which stores active knowledge encoded in back and forward chaining rules.
- Interfaces to external processes (*eg*, structural analysis, CAD).

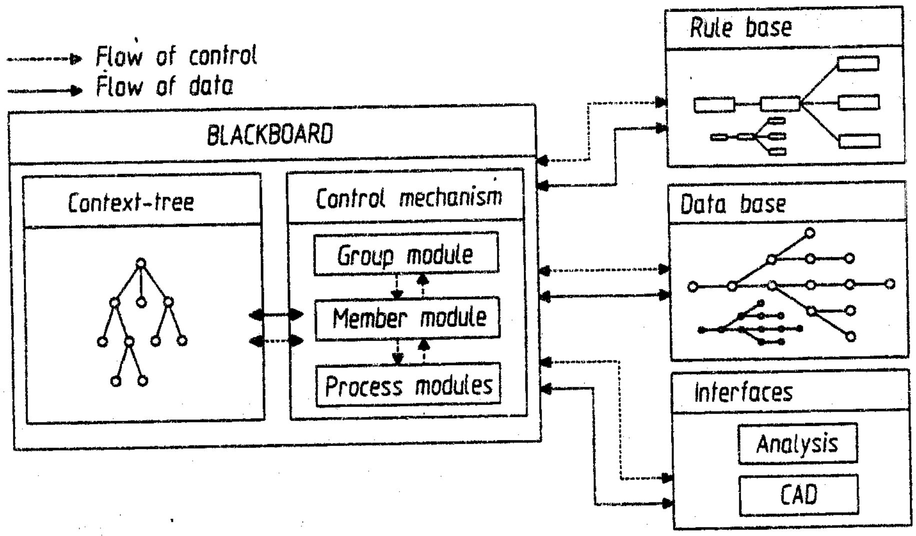

Figure 1: Architecture of the system

Object-Oriented Data Base

The object-oriented data base consists of different knowledge bases or taxonomies. They contain object structures which are necessary to describe the design model. At the present development state of the system three different taxonomies are defined: one to describe the concrete members, one to describe the bending forms of the reinforcement and one to store the material properties of concrete and steel. The focus of the taxonomies is inheritance: superordinated classes are abstractions of subordinated classes and knowledge is left from the first to the second.

During an expert system session the contents of these taxonomies remains unchanged. So the taxonomies represent the passive knowledge of the expert system. As soon as knowledge of a certain object class is needed for the actual design task an instance of the object will be established in the context tree or in the alternative knowledge bases (KBs).

Besides of this passive knowledge the object classes comprise also active methods which perform the data flow between the objects. Methods are programs which represent, along with data storage, the second important aspect of object-oriented programming. Methods do also implement inheritance. For example, the layout of the reinforcement bars is done by a method named "place!". This method is encoded at the class "bar.reinforcement" which is superordinated to all classes representing possible bar reinforcement forms. By the inheritance mechanism the method is left to all subordinated classes and their instances. Only the bars with a special bending form like circular bars have an own specially derived form of the method "place!".

Since KEE is a totally object-oriented system - even the rules are stored as objects in KBs - there are a few more bases in the system. These KBs containing rules or they are serving for the blackboard or for the context tree with its alternatives.

Context Tree

Relations and semantics are the focus of the context tree in contrast to inheritance which is the dominating feature of the object-oriented taxonomies. The context tree construes a kind of working memory of the blackboard and embodies, on one hand, the representation of the design results and, on the other hand, the data structure of the blackboard and its modules. The expert system's activities are focused on the context tree or on the alternative KBs which serve to construct the context tree.

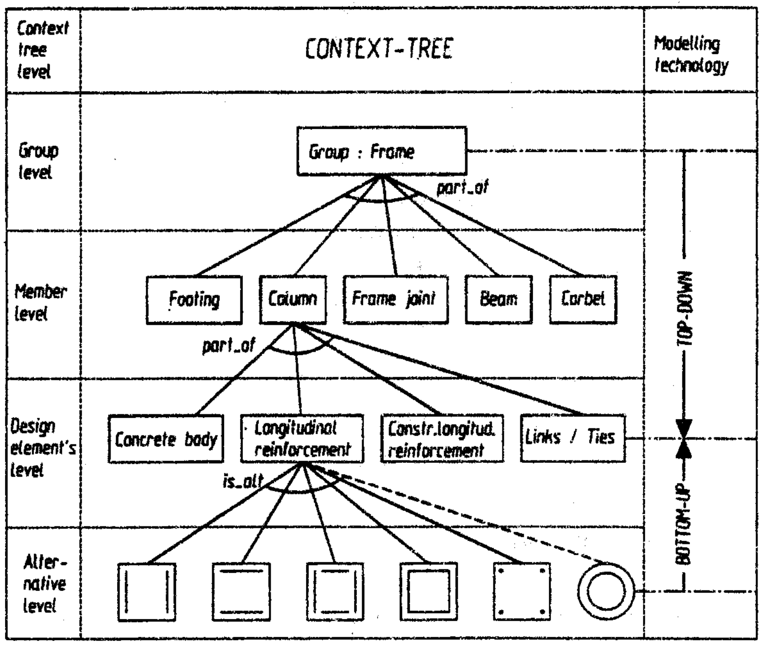

Figure 2: The Context tree

In contrast to the taxonomies where no instances of the object classes exist, the context tree is comprised almost out of instances of the different taxonomies. Only the abstract objects which represent the design elements of the context tree are not instances of any class. They are due to organise and represent the instances of the alternative level (fig. 2).

However, like the taxonomies, the context tree is also organised in a hierarchical form. Yet the higher levels are not only abstractions of the lower levels but the classes on the lower levels represent the components out of which the classes of the higher levels are assembled.

The system consists at the actual state of development out of three components. The LISP based expert system shell KEE (IntelliCorp, 1991) is used as implementation tool for the expert system. As CAD platform the system uses UNICAD (Hochtief, 1991) which is specially developed for the civil engineering domain. To review the reinforcement layouts *ie*, to check whether they conform to the prescribed stress rates, an analysis program is coupled to the system. This program is standard independent because it is not fixed tied to a stress-strain diagram prescribed in a certain standard but allows any stress-strain diagram (expressed by polynomials up to the power of three) to be adapted.

Concept of the System: a Blackboard Model of Problem Solving

One of the main advantages of the classical expert systems is the separation of the knowledge from the inference engine which uses that knowledge to generate a solution for the requested task. However, they have also two weak points:

- "The control of the application of the knowledge is implicit in the structure of the knowledge base *eg*, in the ordering of the rules for a rule based system

- The representation of the knowledge is dependant on the nature of the inference engine (a rule interpreter, for example, can only work with knowledge expressed as rules) (Engelmore, 1989)"

A system with a blackboard architecture is an attempt to overcome these weak points. It implies that knowledge is subdivided in different sources, each equipped with a separate control mechanism. The different knowledge sources communicate by reading and writing in the working memory in a form understandable to all sources, but they must not have the same form or data structure.

The design of the reinforcement of concrete members is affected by different knowledge sources like structural analysis results, standards, design handbooks, geometry, material. Structural analysis is a task which is typically done by algorithmic programs. The representation of standards and handbooks is, on contrary, a task which can be best accomplished by expert system methods.

Because of the various sources a blackboard architecture seems to be advantageous for the implementation of an expert system that must integrate all or most of the sources and processes mentioned above (Garrett, 1987).

The blackboard represents the interface between process modules and programs. It is the place where different processes and modules can communicate by message passing. A special blackboard rule mechanism interprets these messages and by sending respective messages to other processes or modules it controls the solution progress depending on the received message.

The context tree embodies the design strategy of hierarchical planning (Maher, 1989 and Scherer, 1990 a) where a design task is decomposed into smaller elements which can be developed individually considering the interdependencies and the inner relations of the elements. The decomposition of the design task is done in four levels: group level, member level , design element's level and alternative level.

In the group level information about the entire unit, for instance a frame, is stored. Information about the columns, the beams and the joints which build the unit are collected on the member level. Information about the concrete body and the different kinds of reinforcement are concentrated on the design element's level. The objects of these first three levels have a "part_of" relationship to an object of the next level above. The fourth level contains information about the alternatives which are elaborated by the modules of the process model. The objects of this alternative level have an "is_alt" relationship to the objects of the next level above to express that only one of the alternatives can be chosen as solution for a design task on the third level. At the actual state of development of the system only the third context tree level is designed by developing alternatives. In future it is planned to implement a similar design cycle also for the second blackboard level.

To develop parallel different alternatives the necessary parts of the context tree are copied into special alternative KBs where the different alternatives can be developed separately. If one alternative is selected its alternative KB will be copied into the context tree to build the new context for the next design cycle.

Control Mechanism of the Blackboard

The representation of the design progress is modelled in the context tree so that the control task of the blackboard is mainly concentrated on the assemblage of the tree. In accordance to a context tree with four successive levels, the blackboard has three major modules each describing the transition from one level to its subsequent level. The first two modules work top-down. The third module is represented as a process model with a bottom-up technique. The focus of the paper presented here is the third module. For a better understanding the first two modules are also shortly discussed.

The group module develops the first context tree level. It performs the task definition for the unit which the system is going to design. At first the following values are ascertained: determination of the topology, design standards, loads, material values and environmental conditions. Having done this, the module starts the decomposition of the unit (*eg,* a frame) into its members (*eg,* beam, joints, columns and footings). At last the module determines the sequence of treatment for the members.

The member module controls the decomposition of a member into its constructive elements which is the transition from the second to the third context tree level. For example, a column will be decomposed into the following constructive elements: concrete body, longitudinal reinforcement, link reinforcement and, if necessary, constructive reinforcement.

The last module covers the transition from the third to the fourth level of the context tree. The module is configured as a process model with four sub-modules: hypothesis, analysis, evaluation and system selection (Maher, 1984 and Meinecke, 1992). The fundamental alternatives for each design cycle are stored in a design data base (Sriram, 1990). Since this process model is a very important component of the system it is described separately and in greater detail below.

Rule Base

Rules comprise, compared to the more passive knowledge bases, the active part of the system. The design knowledge which is derived from standards and handbooks is represented in rules. One major task of the rules is to check whether the actual design which is represented in the context tree or in one of the alternative KBs conforms to the design standards. Another important task of the rule system is to deduce new facts for the design progress out of already known facts in the context. A third application of rules is the control mechanism for the process model described above.

In general there are two different techniques of rule based programming: forward and backward chaining. The task of each rule is represented in "if - then" scenarios using the object-oriented data structure of the data base. The rules are encoded in TellAndAsk which is the rule component of KEE.

As an example for forward chaining rules the following rule "REINFORCED.3.4A.4B.5C.RULE" representing table 4.2 of the (EC 2, 1992) is given. It determines the concrete cover for the reinforcement of the member. The rule is used in the first blackboard module after the task definition (a question mark is a sign for a locally bounded variable).

```
(reinforced.3.4a.4b.5c.rule                          ;;; mnemotechnic rule name
(if (?member is in class concrete.structure)         ;;; IF, rule premise
  (the type.of.construction of ?member is reinforced) ;;; is ?member a concrete structure ?
  ((the exposure.class of ?member is 3) or           ;;; is the exposure class of ?member 3 or
   (the exposure.class of ?member is 4a) or          ;;; is the exposure class of ?member 4a or
   (the exposure.class of ?member is 4b) or          ;;; is the exposure class of ?member 4b or
   the exposure.class of ?member is 5c))             ;;; is the exposure class of ?member 5c
then                                                 ;;; THEN, rule conclusion
  (change.to (the concrete.cover                     ;;; the concrete cover of ?member is
  of ?member is 4.0))))                              ;;; 4.0 cm !
```

The following backward chaining rule represents the statement that the larger dimension of a column may not be greater than four times the smaller one (EC 2, 1992, §5.4.1).

```
(ec2.col.definition.rule                             ;;; mnemotechnic rule name
(if                                                  ;;; IF, rule premise
  (?alternative is in class rectangular.column)      ;;; is ?alternative a rectangular column?
  (the width of ?alternative is ?w)                  ;;; set ?w to the width of ?alternative
  (the depth of ?alternative is ?d)                  ;;; set ?d to the depth of ?alternative
  (lisp (or   (> ?w (* 4 ?d))                        ;;; is ?w > 4 * ?d or
              (> ?d (* 4 ?w))))                      ;;; is ?d > 4 * ?w
then                                                 ;;; THEN, rule conclusion
  (change.to (the status of ?alternative is false)))) ;;; the status of ?alternative is false !
```

Interfaces

The function of the interfaces is to incorporate external processes with a non object-oriented data structure in the system. The connection is realised by an internal file transfer back and forth from the data structure of the context tree to the data structure of the external process. Two interfaces to external processes are currently implemented. The first connects the CAD system UNICAD with the expert system. At the final state of development it is planned that the user will only communicate with the design system using the CAD platform. The second interface links the already described analysis program to the system.

The Design Cycle: A Process Model for the Representation of the Design Process

As mentioned above the design of the constructive elements is done by a process model with four process modules. The process model itself is domain independent but the rules representing constraints and the classes representing evaluation criteria are of course domain specific. These rules and classes are stored in special KBs. They are dynamically connected to the process model only for the performance of their task. The domain independent functionality of the process model is stored in form of methods and rules in the blackboard.

Hypothesis Process Module

The system contains a data base which covers all useful technical design alternatives for each element of the third context tree level. These fundamental alternatives are mostly represented as methods which perform the initial hypothesis for the alternatives. Each time the system has to perform a design cycle it first looks into the database to find out which and how many alternatives are present for the actual design task. The system depicts a perfect world assumption because it is supposed that the full domain knowledge is available. For example, the alternative layouts for the longitudinal reinforcement of rectangular columns which are implemented into the system are (Grasser, 1979)

a) reinforcement distributed along two sides of the column,
a1) reinforcement distributed along four sides of the column with different steel areas As1 and As2 - only applicable with limited circumstances of load conditions,
b) reinforcement concentrated in the corners of the column,
c) reinforcement distributed along the four sides of the column.

For each possible alternative found in the data base the system creates a special alternative KB into which the required part of the context tree is copied and where the alternative will be developed. The system has the ability to make different alternatives out of one fundamental alternative in varying the value of a characteristic property of the fundamental alternative. For the longitudinal reinforcement layout the system can develop alternatives which are descendants of the above given fundamental layouts but have different diameters and different amounts of reinforcement bars. Note that the initial context remains as long unchanged as the user accepts one of the alternatives as solution for the actual design task.

As soon as the alternatives are created and before they are worked out in details a fundamental mechanical review takes place. Mechanical constraints are checked in a qualitative manner. Alternatives which offend against these fundamental constraints will be eliminated and not further developed. The rule "COL.MZ.ELI.RULE" eliminates alternatives with a wrong load - reinforcement layout condition. Such elimination rules work in the backward chaining technique.

```
(col.mz.eli.rule                                      ;;; mnemotechnic rule name
(if                                                   ;;; IF, rule premise
  (?column is in class rectangular.column)            ;;; Is the shape of ?column rectangular ?
  (all ?long.bars are ?column)                        ;;; ?long.bars belongs to ?column ?
  (the describes of ?long.bars                        ;;; ?long.bars descibes the longitudinal
  is longitudinal.reinforcement)                      ;;; reinforcemnt of ?column ?
  (the strategy of ?long.bars is 2.side.z)            ;;; the layout is only along the z-side ?
  (lisp (= 0 (the ?mz of ?column)))                   ;;; ?mz (bending moment on z-axis) = 0 ?
  (lisp (= 0 (the ?my of ?column)))                   ;;; ?my (bending moment on y-axis) ≠ 0 ?
then                                                  ;;; THEN, rule conclusion
  (change.to (the status of ?long.bars is false))))   ;;; the status of ?long.bars is false !
```

Having done the fundamental review all remaining alternatives are worked out in details *ie*, the instances describing the different alternatives generated by the hypothesis task are created in their special alternative KB.

Analysis Process Module

In this module all alternatives are constrained to conform to the design standards. If one alternative does not satisfy these constraints it will be excluded from the further development of a solution for the actual design task. The quantitative mechanical analysis proving whether the reinforcement and the concrete body observe the stress/strain rates of the design standard is done by an external algorithmic process. The design knowledge is encoded in rules.

An example for a rule implemented in the analysis module is given in chapter Rule Base with the rule "EC2.COL.DEFINITION.RULE". It is used in the design cycle for the concrete body. The analysis rules use the backward chaining technique.

Evaluation Process Module

In order to find the best adapted alternative for the actual design task the system performs an evaluation of the remaining alternatives. The evaluation process requires functionality which is not provided by the rule system. Therefore the evaluation is performed by a domain independent evaluation algorithm (fig. 3) which uses special domain dependent evaluation criteria. For the evaluation process a special KB containing the evaluation criteria is established.

Each criterion is used for evaluating an alternative on two successive context tree levels at least one level must be constrained. Their must be one "main.unit" and there can be one or more secondary units which must be connected to the main unit by a relation given as the first term of the value list (*eg*, "is.used.in"). The second term of the list describes the secondary unit itself. Furthermore, the functionality of the evaluation process is represented in figure 3. An evaluation criterion (*eg*, "DISTANCE.OF.BARS.CLASS") has the following form:

```
distance.of.bars.class                                  ;;; mnemotechnic classname
  main.unit             rectangular.column              ;;; unit to which the criterion must be
                                                        ;;; attached (higher context tree level)
  limit.value           (and (> main.unit depth 40)
                        (> main.unit width 40)
                        (> unit distance.next.bar 30))  ;;; limit value for the criterion (optional)
  ranking               small                           ;;; ranking towards a small or abig value
                                                        ;;; (default: small)
  remarks               economy                         ;;; remarks for the criterion
                                                        ;;; (default: economy)
  slot                  (unit distance next bar)        ;;; slot which shall be constrained
  standard              DIN_1045                        ;;; standard from which the criterion is
                                                        ;;; derived (optional)
  secondary.unit        (is.used.in bar.reinforcement)  ;;; unit(s) which can be constrained
                                                        ;;; (optional) (lower context tree level)
  weight.coefficient    1.6                             ;;; weight coefficient for the criterion
```

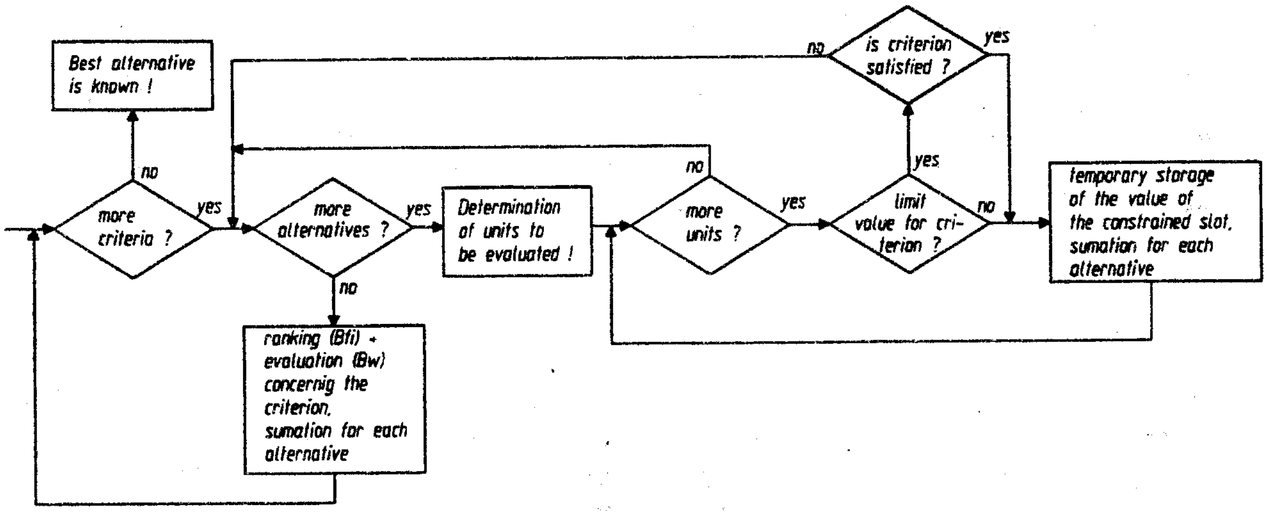
Best alternative is known !
no
more criteria ?
yes
more alternatives ?
yes
no
Determination of units to be evaluated !
ranking (Bfi) + evaluation (Bw) concernig the criterion, sumation for each alternative
no
more units ?
yes
limit value for criterion ?
no
yes
is criterion satisfied ?
no
yes
temporary storage of the value of the constrained slot, sumation for each alternative

Figure 3: Flow Chart of the Evaluation Process

Quantitatively, the evaluation B_w is determined by:

$$B_w = \sum_i c_i \cdot Bf_i \qquad (1)$$

where: c_i= non-dimensional weight coefficient referring to criterion i and
Bf_i= objective ranking factor given by

$$Bf_i = \left| \frac{A_i - B_i}{B_i} \right| + 1 \qquad (2)$$

with: A_i= value of the actual alternative referring to criterion i
B_i= value of the best evaluated alternative referring to criterion i.

The value of the best evaluated alternative B_i is determined depending on the value of the slot "ranking" of each criterion. If ranking is "small", B_i is set to the smallest value of the constrained slot of all alternatives. If ranking is "big", B_i is set to the biggest value. The objective ranking factor Bf_i has the value "1" for the best evaluated alternative referring to the criterion i. For all other alternatives its value is greater than "1".

System Selection Module

As soon as the best adapted and evaluated alternative is determined, the system presents the solution to the user using the CAD platform. If the user does not accept this solution, the system will offer him the second best solution and so on. If the user does accept a solution the system continues with the next design cycle. The KB of the chosen alternative is then copied into the context tree.

Note that each user can adapt the weight coefficients of the evaluation criteria to adjust the system to his own style of design (Mehrafza, 1991). This means that the system can be conditioned. The system selection module can then become superfluous because the system will evaluate solutions using weight coefficient "learned from" the user *ie*, the module would depict a second unnecessary inquiry.

By this adaptation or conditioning the system obtains on the one hand a higher level of automatization but on the other hand it will receive a higher level of acceptance for its automatic generated solutions because their generation is not done in a black box but in an open system where the user can implement his own design ideas.

Conclusion

The choice of a hybrid design expert system with a blackboard approach is a promising step toward the integration of existing programs. The system presented here integrates a commercial CAD system and an existing analysis program. By changing the weight coefficients of the evaluation criteria the user can adapt the system to the way of design he or his company prefers. Such adaptation facilities increase the acceptance of a design expert system.

In the future it would be interesting to perform a backtracking phase to improve alternatives which cannot execute the analysis to make them suitable to the design standards. The full implementation of a CAD interface which performs the whole system-user communication will increase the acceptance of the design expert system since the "normal" engineer may not understand and trust the object-oriented presentation of any design results. Though hard to reach it should be the aim to develop a system adapted to humans and not to any machine or technique.

Acknowledgements

The financial support by the DFG (German Research Foundation) under Contract No. DFG Sche 223/3-1 entitled "Wissensbasierte Bewehrungsfuehrung" (knowledge-based reinforcement design) and through the research project "CAD/CAM Forschungsschwerpunkt" sponsored by the state of Baden-Wuerthemberg is acknowledged.

References

EC 2 (1992), EUROCODE 2 (DIN V ENV 1992 Teil 1-1), *Planung von Stahlbeton und Spannbetontragwerken*, (Design of Reinforced and Prestressed Concrete Structures). Beuth Verlag.

Engelmore R, Morgan T and Nii HP (1989), Introduction. In *Blackboard Systems*, Addison-Wesley, pp 1-22.

Garrett JH and Fenves SJ (1987), A Knowledge-Based Standards Processor for Structural Component Design. In *Engineering with Computers*, 2, pp 219 - 238.

Grasser E (1979), Biegung mit Laengskraft, Schub, Torsion. In *Heft 220: Bemessung von Beton- und Stahlbetonbauteilen,*, (Guideline for Reinforced Concrete Structures) (ed Deutscher Ausschuß fuer Stahlbeton (DAfStb)). Verlag Ernst & Sohn.

Hochtief AG (1991), UNICAD Prgrammierhandbuch, (programmers handbook).

IntelliCorp Inc.(1991), KEE-Manuals.

Maher ML (1984), HI-RISE: A Knowledge-Based Expert System For The Preliminary Structural Design Of High Rise Buildings. *Dissertation*, Carnegie-Mellon University.

Maher ML (1989), Structural Design By Hierarchical Decomposition. In *Computing in Civil Engineering, Proceedings of the Sixth Conference, ASCE*, pp 195-202.

Mehrafza MJ and Scherer RJ (1991), Regelbasierte Formulierung von Variantenelementen in der Stahlbetondetaillierung, (Rule Based Formulation of Parameterized Elements in Detailing Reinforced Concrete Buildings). In *Proceedings of the "Dortmumnder Expertensystemtage"* (ed K Heinz). Verlag TUEV Rheinland, pp 274-287.

Meinecke C and Zlajpah Z (1992), Object-Oriented Programming to Include Standard Knowledge into Expert Systems. In *Computer in Civil Engineering, Proceedings of the Sixth Conference, Ljubljana Slovenia* (ed J Duhovnik), pp 302-307.

Scherer RJ (1990 a), Expertensysteme fuer die Planung und Bemessung der Bewehrung von Stahlbetonbauteilen, (Expert Systems for the Design of Reinforced Concrete Structures). In *CAD/CAM - Auf dem Weg zu einer branchenuebergreifenden Integration* (ed R Anderl and P Castro), Springer Verlag, pp 200-215.

Scherer RJ (1990 b), Formen der Wissensrepraesentation in einem Expertensystem fuer die Stahlbetondetaillierung, (Different Kinds of Knowledge Representation in an Expert System for Detailing Reinforced Concrete Structures). In *KI-Forschung im Baubereich* (ed J Gauchel), Verlag Ernst & Sohn, pp 171-182.

Scherer RJ and Katranuschkov P (1993), Architecture of an Object-Oriented Product Model Prototype for Integrated Building Design. To apear in *Proceedings of the 5th Int. ASCE Conference on Computing in Civil Engineering.*

Sriram D, Cheong K and Kumar ML (1990), Engineering Design Cycle: A Case Study and Implications for CAE. *Research Report No. R91-03.* Massachusetts Institute of Technology.