INTELLIGENT DESIGN TOOLS AS PRODUCT MODEL INTERFACES

Hauser M.[1], Nollau C.[1] and Scherer R.J.[2]

ABSTRACT: Concurrent engineering is understood as a methodology that can be used to increase quality and reduce design effort and time-to-market of a product. Main aspects of concurrent engineering are information sharing among multiple design agents and problem solving actions. Associated techniques are product models and intelligent design tools. In the paper the interrelationship of product modelling technology and intelligent design support is reviewed.

A practical contribution to concurrent engineering by using design tools as intelligent interfaces to product models has been done by the authors. The paper describes the concept and prototypic implementation of two knowledge-based design tools in the domain of structural design. The tools form integrated parts of a design environment linked via a central product model.

KEYWORDS: Concurrent Engineering, Intelligent CAD, Product Modelling

1 INTRODUCTION

There is nowadays a broad consensus that product modelling technology is essential for future computer aided engineering environments. Various product modelling frameworks and approaches exist. Traditionally product modelling research focuses on issues concerning the translation of information between different CAD/CAM systems, interoperability and conflict management. The question how the data that is input into a product model is generated, is often neglected in favour of data management issues.

On the other hand it seems obvious that when the integrated management of high-level product data becomes achievable the design agents should be capable of using the inherent potential of product descriptions with rich semantics. The best advantage of advanced product modelling techniques can be taken when intelligent tools exist that allow the transfer of technical knowledge and specifications via user-friendly interfaces into the product model.

2 PRODUCT MODELLING

The idea behind using product models in structural engineering design is to cope with the complexity of the development of integrated environments for design. Generic uniform interfaces for the design tools are necessary. In that sense product models schemes should serve as a common language for the interchange of design information.

Product models are information models that define and instantiate product information that is relevant to more than one design agent in the scenario. Building blocks of product models are descriptions of design objects, design relations and constraints. By defining a common set of modelling schemes and an associated syntax the integration of multiple design agents can be simplified.

[1]Research Assistant, Institute of Structural Mechanics and Applied Computer Science, Dresden University of Technology

[2]Professor, Institute of Structural Mechanics and Applied Computer Science, Dresden University of Technology

2.1 ISO-STEP for structural engineering

One of the most well known standardisation approaches in product modelling is the ISO 10303 (STEP) standardisation effort (ISO TC184/SC4 1994). STEP is dedicated to the implementation of back-end translators between existing CAD systems. Application protocols (AP) determine how the integrated resources defined in the basic parts should be used in supporting a particular application. STEP standardisation especially relevant to structural engineering design is the application protocol 225. AP 225 covers the representation of shapes of structural building members such as walls, beams, columns and slabs.

Common to the actual approaches is that the goal of providing common representation facilities for design entities, their relations and associated constraints has not yet been fully met for structural engineering design. When refering to structural engineering the above approaches are to date restricted to modelling issues for geometry and topology. The exchange of complete design semantics is not yet supported.

In distributed concurrent engineering environments data exchange using files is a common technical solution. The exchange file format defined by STEP part 21[3] of STEP does not provide a mean to solve the antagonism between exchanging semantically rich and thus highly interrelated entities and incremental data exchange. There is no facility to formally define a given context for a partial exchange.

When considering the objective of STEP to provide means for translating information between *existing* CAD systems the modelling shortcomings are also a consequence of today's design practise. Mostly the design practise is still on the level of 2D drafting.

We claim that in order to introduce concurrent engineering in structural engineering design the product modelling, data exchange and intelligent design support technology of today is not sufficient. There is a gap between nowadays design tools and data exchange frameworks on the one side and the requirements of concurrent engineering in structural engineering on the other side.

We propose that in addition to the actual standardisation activities for structural engineering and related efforts a new level of design support and associated data exchange facilities is necessary. The development of product modelling schemes for structural design with richer semantics should be initiated by developing corresponding design agents and assistance systems that can deal with semantics besides geometry.

2.2 Intelligent CAD and product modelling

To investigate design reasoning and associated semantics the research direction *Intelligent CAD* has been established already about 30 years ago. Intelligent CAD deals with theoretical and practical issues of design knowledge representation and problem solving activities (Akman, ten Hagen and Tomiyama 1990, Tomiyama and Yoshikawa 1987, Forbus 1988). We consider research in this field to be the potential originator for semantically rich data exchange and information sharing in concurrent engineering environments for structural engineering design.

Research in intelligent CAD is traditionally oriented towards mechanical engineering. Compared to the area of mechanical engineering the efforts dedicated to design in the building industry are relatively rare (Maher and Fenves 1985, Giretti, Spalazzi and Lemma 1994, Drach 1994, Zhang and Maher 1993, Flemming 1990, Luth, Krawinkler and Law 1991, Hauser, Nollau and Scherer 1995).

[3] Part 21 “Clear text encoding of the exchange structure” is one of the parts of STEP that deal with implementation methods, i.e. specify how STEP application protocols will map on to real software.

To enable advanced product data exchange as necessary for concurrent engineering in structural design there is a backlog for intelligent CAD research in the building industry sector. Present CAD tools for structural design do not directly support the semantics of the designer's internal models. Their modelling is often restricted to geometry and the user has to reduce and map his models on a set of geometric primitives.

The output of the different design stages often remains to be paper-oriented. Collaboration and integration demand re-usable and machine readable data structures and output. Design systems should allow the export of the design results and associated background knowledge for further processing and use by other programs. Explicit representation and handling of design relations, constraints inside the design tools is a prerequisite. It can not be expected that the designers participating in a concurrent engineering environment create and manage the complex data structures related to design of the product building “by hand”. Thus persistence and exchange of design semantics are necessary across multiple tools.

A first step to full-fill the objective target to deal with design semantics in concurrent engineering is to set up a framework for intelligent design support. This framework should be especially suitable to structural engineering design.

3 INTELLIGENT DESIGN TOOL FRAMEWORK

In the next sections we discuss the concept and prototypical implementation of work of the authors that is on the one hand side seen as a contribution to intelligent CAD in the field of structural engineering design and on the other hand side considered to be valuable preparatory research in the field of product model interface technology for concurrent engineering.

3.1 Design product model

The model of the artifact that is to be designed is given by a set of interrelated design instances. Design instances represent instantiated models of the domain's entities. In the context of structural design this includes architectural elements and structural members and systems. A design Model $M_{I,R}$ is given by a set of design instances I and the set of relations R between these instances.

3.1.1 Design instances

We use the terms instances and object class according to the commonly understood meaning. Instances are structured units of attributes (slots) and assigned values. Instances belong to one or more object classes. Object classes represent categories of things and concepts. Attributes and methods may be directly attached to an object class or may be inherited from one or more superior classes.

3.1.2 Design relations

We take an approach that explicitly models the relations between design instances. Relations are essential for interpretation and consistent modification of a design model. An explicit representation of relations is most effective to deal with the evolution of the design model. Relations reflect semantic connections between design instances. Using an extensional description a binary relation[4] R is described by an associated predicate p_R: I_1 is related by R to $I_2 \Leftrightarrow p_R(I_1, I_2)$. Most common relations are of sub-types *aggregation*, *composition* and *specialisation*.

[4] *1-to-n or n-to-m* relations can be constructed with multiple separate binary relations.

3.2 Design reasoning

Since design is a cognitive process the most prominent aspect is the involved kind of reasoning. Design mainly involves reasoning in the theory associated with the domain's design objects.

Cooperative work in a concurrent engineering environment demands that a given design agent is able to classify the casual context and the intention of other cooperating design agents. Thus we consider an explicit formalisation of different reasoning levels as essential.

For design task related to structural engineering we have identified three levels of reasoning[5]:

Strategic level Decisions of general nature are made and the abstraction level is high. Reasoning on this level remarkably involves projection and anticipation. An example for a strategic decision is to build a structural system as a skeleton.

Tactic level The degrees of freedom are limited by corresponding strategic decisions. The used models of the domain objects are increasingly detailed. In our modelling framework the concept of dedicated tools is used to model tactic design actions. Tactic design actions relate to the assignment of values to a focused set of design parameters. Design actions are embedded in the context of previously committed and further anticipated actions. Tactic action are e.g. selection of field-lengths for a floor system among several alternatives.

Reactive level On this level instances and models are rather detailed. Actions can be interpreted as reactions to other actions or as reactions to the presence of given types of design instance configurations. A reaction may be for example to re-dimension a given structural member because the capacity of its cross-section proves to be exceeded.

The design process can be interpreted as a search for feasible design solutions within a design space. In general complete search as a “brute force” approach is not possible because of the complexity of the design space. Corresponding to our terminology of strategic, tactic and reactive design levels on each of those levels distinct knowledge to direct and constrain the design search is used. Each level provides an own granularity of search steps in the design space. In that context design strategy can be interpreted as anticipation of a promising search path through the design space. Design tactic can be seen as a selection of search nodes. Reactivity refers to local modification of search nodes. In general these local modifications are assumed to be compatible with the tactic or strategic decisions.[6]

3.2.1 Design approach planning (strategy level)

The strategic design reasoning level deals with anticipations and expectations. Therefore reasoning does not relate to concrete design instances but to abstractions and facts in form of features. This includes the features present in the current design state as well as those anticipated to be generated by the design approach. Feature descriptions are a priori defined in parametric schemes. Control knowledge for the design process is represented in the design operators used for the planning of the design approach.

Design operators represent parameterised types of design steps. An operator specification defines predicates when a given operator type is applicable and the transformation to the design state that is expected. This transformation is defined with a set of new features that are expected. A design operator $O_{C,F}$ is given by a set of conditions C that define its applicability and a set of features

[5] These levels might as well apply to other domains but that is beyond our current scope.

[6] Nevertheless if the modifications prove to be non-local propagation steps have to be triggered. Propagation of changes is a procedure associated with the reactive design reasoning level.

F that are assumed to be present after the operator is applied. An operator is applicable when all conditions in C are satisfied by the set of actual present features F. Conditions may be of simple type, disjunctions or quantified.

When a design operator is applicable there is in general a set of valid parameter bindings. A design operator that is instantiated with a selected parameter substitution Θ is written as $op_{C,F,\Theta}$. When a design operator instance is integrated in the planned design approach the related features are anticipated to be present in the model after the actions corresponding to the design operator have been committed, i.e. $O^t_{C,F,\Theta} \Rightarrow t\alpha(\forall f \in F : M_{I,R} \vdash f_\Theta)$. The form $t\alpha p$ denotes the fact that the predicate p is assumed to be satisfied after time t.

The time-mark t_i that is associated to a design operator instance $op^{t_i}_{C_i,F_i,\Theta_i}$ during the planning phase of the design approach is a virtual mark used to define a total ordering for the sequence of operators. When an approach is committed, i.e. corresponding design actions on the tactic level are instantiated, the time mark t_i is mapped on a real time t^r_i denoting the actual time of execution.

A design approach is a sequence of operators op_i. Design reasoning on the strategic level can be interpreted as the problem to find at least one valid design approach for a given model state $M_{I,R}$ and a set of desired conditions C. A design approach is valid when: $op^{t_1}_{C_1,F_1,\Theta_1} \circ op^{t_2}_{C_2,F_2,\Theta_2} \circ \ldots \circ op^{t_n}_{C_n,F_n,\Theta_n} \Rightarrow \forall c \in C : satisfied(c, F_1 \cup F_2 \cup \ldots \cup F_n)$.

In this interpretation the problem to find a design approach is a classical planning problem. AI research has invented different planning algorithms and paradigms. Since in the actual state of development we assume that we do not have to handle uncertainty in planning we use the classical goal-oriented planning approach of AI. We do not consider the above assumption to be critical since planning in our system architecture is deliberately situated on the level of expectations. Corrections and adaptions can be carried out on the tactic or on the reactive level.

The design focus comprises the set of actual identified possible design approaches and the focused operator instances in these approaches. For a given design approach exactly one focused operator instance exists. The focused operator instance for a design approach is the operator instance following the latest applied operator instance. Focused operator instances can be integrated in the design path. The operators are hierarchically organised, i.e. it may be necessary to expand abstract operator instances in sub-approaches.

We distinguish between primitive and expandable design operator schemes. A primitive operator expands the operator notion $O_{C,F}$ with a set of applicable tools T to $O_{C,F,T}$. Tools model design actions (see description of tactics level below).

An expandable operator $O_{C,F,C_{ex}}$ specification is extended with a set of conditions C_{ex} that serve as goals for a hierarchical expansion of the design approach. The integration of an expandable operator instance in the design path triggers the generation of approaches for the sub-problem defined by the expansion goals C_{ex} and thus a new design focus is generated. Design reasoning continues on the strategy level.

The integration of a primitive operator leads to the execution of design actions. Reasoning is then displaced to the tactic level.

3.2.2 Design actions (tactic level)

When suitable completed design approaches are available the designer proceeds along these anticipated design paths by committing the associated foreseen actions.

On the strategic level expectations about a transformation of the design model that takes place when corresponding actions are applied have been specified by the design operators. Design actions actually

cause a change of the design model.

A design action is:

1. add a design instance i. This is a model transition $M_{I,R} \rightarrow M_{I \cup ihull(i), R \cup rhull(i)}$
2. evolve a design instance i to i'

$$i = \{ a_j \equiv \begin{cases} slotvalue(a_j, i') & \exists a_i \in \text{A}ttributes_{i'} : a_i = a_j \\ slotvalue(a_j, i) & \bot \end{cases} \}$$

3. remove a design instance i. This is a model transition $M_{I \cup ihull(i), R \cup rhull(i)} \rightarrow M_{I,R}$

When adding or removing design instances it has to be considered that domain specific dependencies between classes of design instances exist, e.g. the plate and the beams as parts of a structural plate system depend on the aggregated design instance plate-system. The above used function $ihull(i)$ computes the transitive hull of design instances dependent on the instance i. The function $rhull(i)$ denotes all relations related to the instance i or its transitive dependents.

To commit design actions we use the notion of tools. Tools model design actions. Design actions actually modify the design model state $M_{I,R}$. The procedural knowledge that is involved when actually committing a design step is defined by tool specifications.

A tool specification T consists of an interaction part defining the iconic representation in the user interface, a predicate p_T for the validity of target objects, furthermore a constructor C_T for instantiating or evolving design objects, a set V_T of possible variations of constructor parameters and a credit assignment function cr.

When $TA = \{ta_1, ta_2, \ldots ta_n\}$ is the set of possible targets that have been identified by user interaction then a target ta_i is valid if $p_T(ta_i)$ is true. For a chosen valid ta_i the application of C_T to each tuple $(ta_i, Para_j)$ with $Para_j$ being a variation of parameters derived from V_T, results in a set of alternatives $\{a_1, a_2, \ldots, a_n\}$. The alternatives a_i are presented to the user for selection ranked according to the credit assignment $cr(a_i)$.

Tools appear to the user as icons in a palette of available tools. By dragging the icons on the representation of design instances in graphical user interface the user can manipulate the corresponding design objects, e.g. commit dimensioning or system definitions. As building blocks for the definition of the constructors and evaluation function serve the general structural mechanics computation functions defined in an object library specific to the structural design domain.

A tool application has a feedback to the strategic reasoning level. All design approaches that are not compatible with the given tool application are removed from the set of focused design approaches. A design approach with a focused instance of the operator scheme $O_{C,F,T}$ is compatible to an application of tool t if $t \in T$.

3.2.3 Design modifications (reactive level)

Design actions are deliberately taken and correspond to the anticipated design approach strategy. They result from strategic and tactic reasoning. Design modifications are situated on the reactive reasoning level. Design modifications result from direct manipulation of design instances by the user or may also occur when pre-defined constraints are violated by design actions.

Constraints are used for the detection of conflicts. In our approach we do not use constraints for synthesis purposes or for the computation of possible solution sets. Design constraints are attached to domain object classes and a given design context, i.e. the application of a design action. That means

for each instance of the specified class the constraint is instantiated and checked every time when the specified design action is applied.

A design constraint $co_{cl_o,test,pa,con,R}$ defines an owner class cl_o, a predicate *test*, a relevance context *con* and a set of reactions *R*. The constraint $co_{cl,test,p,con,act}$ is attached to an instance *i* when the class of *i* is subsumed by the class cl_o. When *i* is involved in design actions in the context *con* the predicate $test(i, pa(i))$ is evaluated. The function *pa* defines possible conflict partners for *i* for the given constraint. Typically *pa* makes use of the design relations of *i*. If *test* evaluates to true a set of possible reactions $R_{i,pa(i)}$ is instantiated. The elements of $R_{i,pa(i)}$ are presented to the user for selection. Each $m \in R_{i,pa(i)}$ represents a design modification.

A design modification is a modification of the attribute values of a design instance. Design modifications are recursively propagated from the design instances to which they have been applied to related instances that should also be modified to keep the design model consistent. For this purpose propagation rules can be specified.

A propagation rule $r(S, M, succ, history)$ is given by the design instance class *S*, a modification type *M* and the functions f and *succ*. If a modification m_i to instance *i* occurs the rule $r(S, m, f, succ)$ is applicable when the type of m_i is subsumed by *m* and the design instance class of *i* is subsumed by *S*. If the rule is applicable then after the modification is committed and propagated to the design instances computed by a propagation function *succ*. In consequence additional propagation rules may be applicable.

3.3 Exchanging design semantics

In the design framework we have envisioned the semantics related to the design process and the artifact. We focused on the different levels of reasoning. When a common understanding of the reasoning levels exists, then besides pure product data also associated design strategies, tactics for the actual approach, change propagations and expected or present design features can be exchanged between multiple design agents. This will enable true concurrent engineering support.

In the next section we discuss prototypic implementations of design agents aligned with the discussed framework.

4 PROTOTYPIC IMPLEMENTATIONS

Two prototypes of intelligent design tools have been implemented in the domain of preliminary structural design and in the domain of reinforcement design on the basis of a generic framework that has been derived from the above outlined considerations. The prototypes have been implemented based on the *Generic INteractive Application* (GINA) framework (Spenke and Beilken 1990) in the CLOS object system of COMMON LISP (Keene 1989).

4.1 Preliminary structural design

This section introduces the design system for preliminary structural design developed in respect to our paradigms for intelligent CAD tools (Hauser and Scherer 1995, Hauser and Nollau 1995). The design system serves as an assistant to the designer (architect, structural engineer) and helps to derive early conclusions on the structural system of a building and the dimensions of its main members. The quality and correctness of these first assumptions and kernel ideas govern the extend of corrections that are necessary for the exact structural calculations for the final design and thus is essential for the effectiveness and quality of the design as a whole.

The preliminary structural design prototype has been used and demonstrated in the ESPRIT project COMBI. (Scherer 1995) A description and user manual is available (Hauser et al. 1995). The knowledge base of the system contains operators and tools that define sequences and alternatives for (partial) system definitions and member dimensioning.

Figure 1 shows a snapshot of the desktop of the system[7].

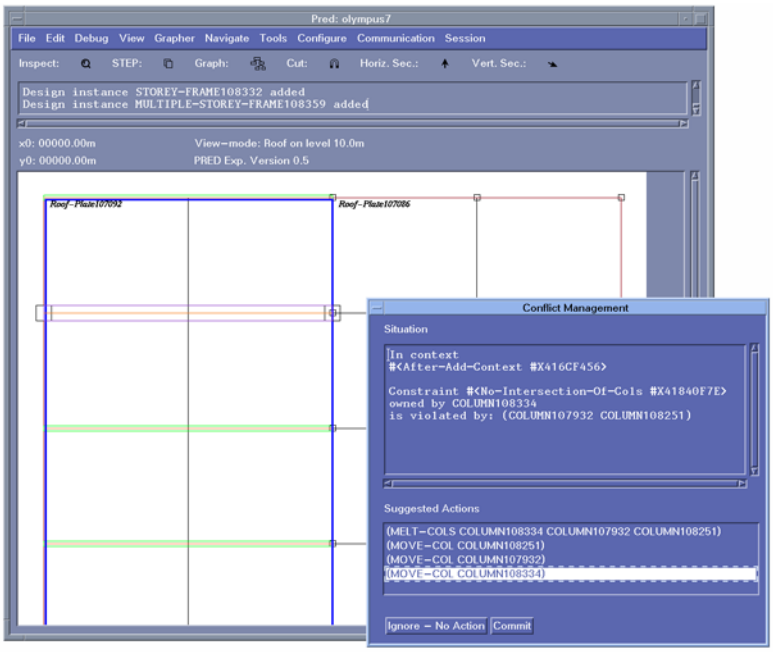

FIG. 1: Preliminary structural design application desktop

4.2 Reinforcement design

To date a second prototype is implemented for the domain of reinforcement design. For common standard reinforced concrete (partial) systems – e. g. slabs, beams and columns – many software tools which support the design from the structural analysis to the CAD drawing of the reinforcement system are already available. But these existing solutions are only applicable to for those structural elements, which can be calculated with simple rules. In the reinforced construction of a building as a whole there are many critical construction details which need special dimensioning rules and a lot of engineering knowledge. Especially the transition from design data concerning assumptions on the reinforcement placement to resist bending moments, shear and compression forces to the correct layout of reinforcement needs a lot of experience in construction and technology of this process. With a prototypical application for continuous girders we test the application of our conception for intelligent CAD tools in the field of reinforcement design. The design process captured involves calculation of the required reinforcement, reinforcement layout and instantiation in the girder fields.

Due to a common paradigm for interaction and design support the desktop of the reinforcement design system (figure 2) appears similar to the above shown desktop of the preliminary design system. The difference is that geometric navigation and viewing is rather organised in element sections, girder fields and reinforcement layers.

[7]The snapshot shows a situation in which an instance of the constraint scheme 'No-Intersection-Of-Cols' is violated and recovery actions are suggested.

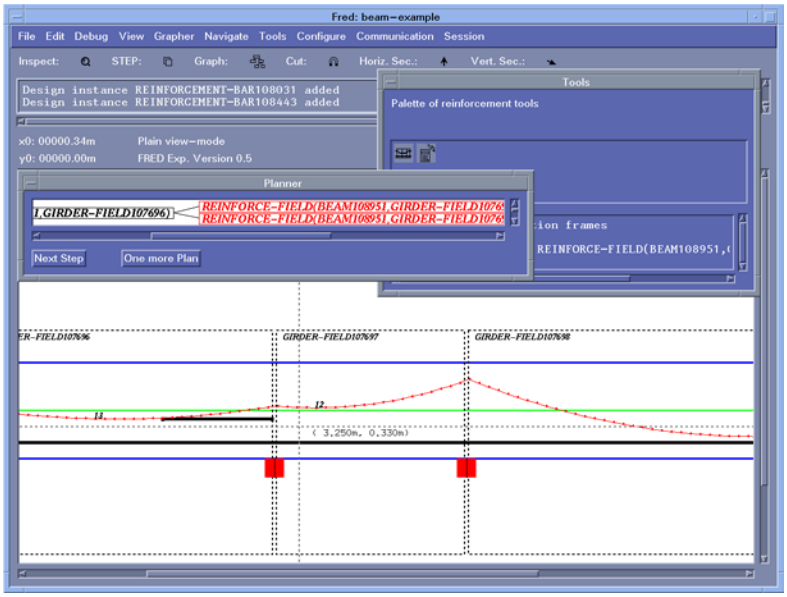

FIG. 2: Reinforcement design application desktop

4.3 Prototype integration scenario

The preliminary structural design assistant system has been integrated in the product modelling framework of the ESPRIT project COMBI[8](Scherer 1995).

The COMBI prototype scenario yields a methodology which is ready to serve as a platform of network-based design in a virtual enterprise. The scope of COMBI is the management and control of product data. The COMBI methodology is already prepared for an extension to concurrent engineering. The developed integration tools provide means for process control, consistency maintenance of the product data base, object management, model mapping – i.e. transformation of the data connected with one application to another – communication with remote applications via the Internet.

5 Conclusions

We claimed that concurrent engineering in the structural engineering domain needs more sophisticated data exchange facilities. We related the lack of such advanced data exchange means to the shortage of intelligent design tools in the structural engineering domain. To overcome this shortage we developed a conceptual framework for intelligent design support. The generic framework has been successfully specialised for the two application prototypes that have been briefly presented.

The framework is not related to a certain software organisation and programming logic but rather defines reasoning and model structures we consider essential for interactive intelligent systems for CAD. We believe that on the basis of these structures also an advanced framework for exchanging design semantics is feasible.

References

Akman, V., ten Hagen, P. J. W. and Tomiyama, T.: 1990, A fundamental and theoretical framework for an intelligent CAD system, *Computer-Aided Design*.

[8]COMBI is an acronym for "**C**omputer **O**riented **M**odels for the **B**uilding **I**ndustry".

Drach, A.: 1994, *Flexible Werkzeuge für die integrierte Gebäudeplanung*, Vol. 125 of *Fortschrittsberichte VDI, Reihe 4: Bauingenieurwesen*, VDI Verlag, Düsseldorf.

Flemming, U.: 1990, Eine Hülle zur Entwicklung von wissensbasierten Systemen für die Grundrišplanung, *in* Gauchel (ed.), *KI-Forschung im Baubereich*, Ernst & Sohn Verlag, Berlin.

Forbus, K.: 1988, Intelligent computer-aided engineering, *AI Magazine*.

Giretti, A., Spalazzi, L. and Lemma, M.: 1994, A.S.A. an interactive assistant to architectural design, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '94*, Kluwer, Dordrecht, Netherlands.

Hauser, M. and Nollau, C.: 1995, A tool to support preliminary structural design of industrial buildings (in German), *in* V.Berkhahn and M.Olbrich (eds), *Forum Bauinformatik: Junge Wissensschaftler forschen*, Vol. 173 of *Fortschrittsberichte VDI, Reihe 20*, VDI Verlag, Düsseldorf.

Hauser, M. and Scherer, R. J.: 1995, Knowledge acquistion and representation for preliminary structural design, *in* P. J. Pahl and H. Werner (eds), *Proceedings of the 7th International Conference on Computing and Building Engineering*, Balkema Publishers.

Hauser, M., Nollau, C. and Scherer, R. J.: 1995, Expert system for preliminary design: Documentation of the prototype, *ESPRIT Project 6609 - COMBI, WP D, Deliverable 4*, Lehrstuhl für Computeranwendung im Bauwesen, TU Dresden, Germany.

ISO TC184/SC4: 1994, ISO 10303-1: Industrial automation systems and integration – Product data representation and exchange – part 1: Overview and fundamental principles, *Technical report*, International Organization for Standardization.

Keene, S. E.: 1989, *Object-oriented programming in COMMON LISP - A programmer's guide to CLOS*, Addison-Wesley.

Luth, G. P., Krawinkler, H. and Law, K.: 1991, Representation and reasoning for integrated structural design, *Technical Report 55*, Center for integrated facility engineering (CIFE), Standford University.

Maher, M. L. and Fenves, S. J.: 1985, HI-RISE: An expert system for the preliminary structural design of high rise buildings, *in* J. S. Gero (ed.), *Knowledge-Engineering in Computer-Aided Design*, Elsevier Science.

Scherer, R. J.: 1995, COMBI: Overview and objectives, *in* R. J. Scherer (ed.), *Proceedings of the First European Conference on Product and Process Modelling in the Building Industry*, Balkema, Rotterdam.

Spenke, M. and Beilken, E.: 1990, An overview of GINA, *in* D. A. Duce et al. (eds), *User Interface Management and Design*, Springer-Verlag, Berlin.

Tomiyama, T. and Yoshikawa, H.: 1987, Extended general design theory, *Design Theory for CAD*, North-Holland.

Zhang, D. M. and Maher, M. L.: 1993, Using case-based reasoning for the synthesis of structural systems, *Knowledge-based Systems in Civil-Engineering*, IABSE Colloquium.