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

Artikel: Preliminary design of bridges using knowledge-based systems
Autor: Edlund, Bo / Löfqvist, Per
DOI: https://doi.org/10.5169/seals-51858

Preliminary Design of Bridges Using Knowledge-Based Systems

Avant-projet de ponts à l'aide de systèmes experts

Vorentwurf von Brücken mit Hilfe von Expertensystemen

Bo EDLUND
Professor
Chalmers Univ. of Technology
Gothenburg, Sweden

Per LÖFQVIST
Res. Assistant
Chalmers Univ.of Technollogy
Gothenburg, Sweden

Bo Edlund, born 1936, received his Civil Eng. degrees at Chalmers Univ. Since 1967 teaching and research in Struct. Eng. his special research interests are stability problems, thin-walled structures, CAD, timber struct.

Per Löfqvist, born 1961, got his M.Sc. in Civil Eng. from Chalmers Univ. in 1987 and his degree of licentiate in Eng. in 1992. He has been teaching both in Struct. Eng. and in the Artificial Intelligence domain.

SUMMARY

This paper concerns methods for the application of knowledge-based techniques to preliminary structural design. Different methods which can be used to improve software for computer aided design and engineering are discussed. A practical study of these methods is made by developing a prototype for preliminary design of highway bridges. In order to allow the design engineer to test different alternatives and to see the consequences of a certain design choice or design step a method which enables the user to make assumptions has been implemented. The use and the suitability of product models in computer aided structural design is also discussed.

RÉSUMÉ

L'exposé se réfère à l'application des techniques de systèmes experts dans l'avant-projet de structures porteuses. Il présente diverses méthodes en vue d'améliorer les logiciels de conception et de construction. Une étude pratique de ces procédés a permis de développer un progiciel utilisable dans le prédimensionnement des ponts routiers. En vue de fournir à l'ingénieur projeteur la possibilité d'examiner des alternatives, ainsi que les conséquences de choix effectués lors de l'avant-projet, ce progiciel comporte une méthode permettant d'entrer les hypothèses envisagées. L'article examine l'aptitude de produits modèles à être utilisés dans le projet assisté par ordinateur.

ZUSAMMENFASSUNG

Der Beitrag behandelt die Anwendung von Expertensystemtechniken auf den Vorentwurf von Tragwerken. Verschiedene Methoden zur Verbesserung von CAD/CAE- Software werden diskutiert. Ihre Anwendbarkeit konnte bei der Entwicklung eines Prototypprogramms für die Vorbemessung von Strassenbrücken erprobt werden. Um dem entwerfenden Ingenieur zu ermöglichen, verschiedene Alternativen und die Auswirkungen bestimmter Entwurfsentscheidungen zu studieren, enthält das Programm eine Methode zur Eingabe von hypothetischen Annahmen. Weiterhin wird die Verwendbarkeit von Produktmodellen in CAD für Tragwerke angesprochen.

1. INTRODUCTION

Common for all CAD/CAE–programs of today is that they are used at a relatively late stage of the design process, e.g. for the analysis and documentation of a "known" structure. Programs that support the early stages of design are rare and, if they exist, very specialized. Future generations of CAD/CAE–programs must be able to support the early stages of design. In order to accomplish such a support future CAD/CAE–programs must be able to describe knowledge such as codes, experience, heuristics etc. The use of expert system techniques is one way to deal with such knowledge.

Future generations of CAD/CAE–programs will also be working in a more complex environment than the CAD/CAE–systems of today and must be able to exchange data and information with external sources such as databases. In the future it is likely that for computer aided design there will be just one shared database for each project. Different computerized tools will use this shared database to retrieve and store information. One purpose of this approach using just one shared database is to avoid multiple storage of one and the same information. A database, that can be used during the entire design process, will be complex and must contain so much information that it should be able to cover all relevant aspects of the current problem field. An object description, which includes all types of information needed to describe the object during the entire design process, is often called a *product model*. This paper discusses the use of product models in computer aided structural design. It is also discussed how a knowledge based system can be devised by combining product models with production rules and/or procedures.

A practical study of these methods is made by developing a prototype (called PREBRI) for preliminary design of highway bridges. In order to allow the design engineer to test different alternatives, and to observe the consequences of a certain design choice or design step, a method which enables the user to make assumptions (hypotheses) has been implemented in PREBRI.

2. KNOWLEDGE REPRESENTATION IN DESIGN

2.1 Knowledge representation and databases

In structural design of today we use several different methods to represent knowledge. It can be visualized as *sketches, drawings, flow diagrams, results from analyses* etc. Different methods describe different aspects that designers have on the structure (figure 1). Each method gives a relevant description of the designed object used under different circumstances or by different actors in the design process. These methods have been developed under a long period of time and fulfil the needs that engineers, architects and contractors have in the process of building design. Different actors in the design process do not in general have difficulties in understanding these methods and to separate them from each other.

Computer programs for structural design purposes used today tend to reflect the above–mentioned methods to represent knowledge. There are programs for drafting, analysis, project planning etc. Computer programs used to solve a particular problem are usually effective tools and fulfil the task for which they were designed. Computer programs cannot (in contrast to humans) understand different methods to describe knowledge. This makes it difficult to exchange information between different kinds of programs. It is for example hard to use a drawing created in a CAD–program as input to a FEM–program and vice versa. A FEM–program cannot by itself extract those parts of a drawing that are necessary for the structural analysis. It can sometimes even be difficult to transfer information between different computer programs designed for the same task, for example to transfer a drawing from one CAD–program to another. In the latter case we often risk to lose information because the drawing has to be transferred using a standard, which just includes a minimum set of graphical information.

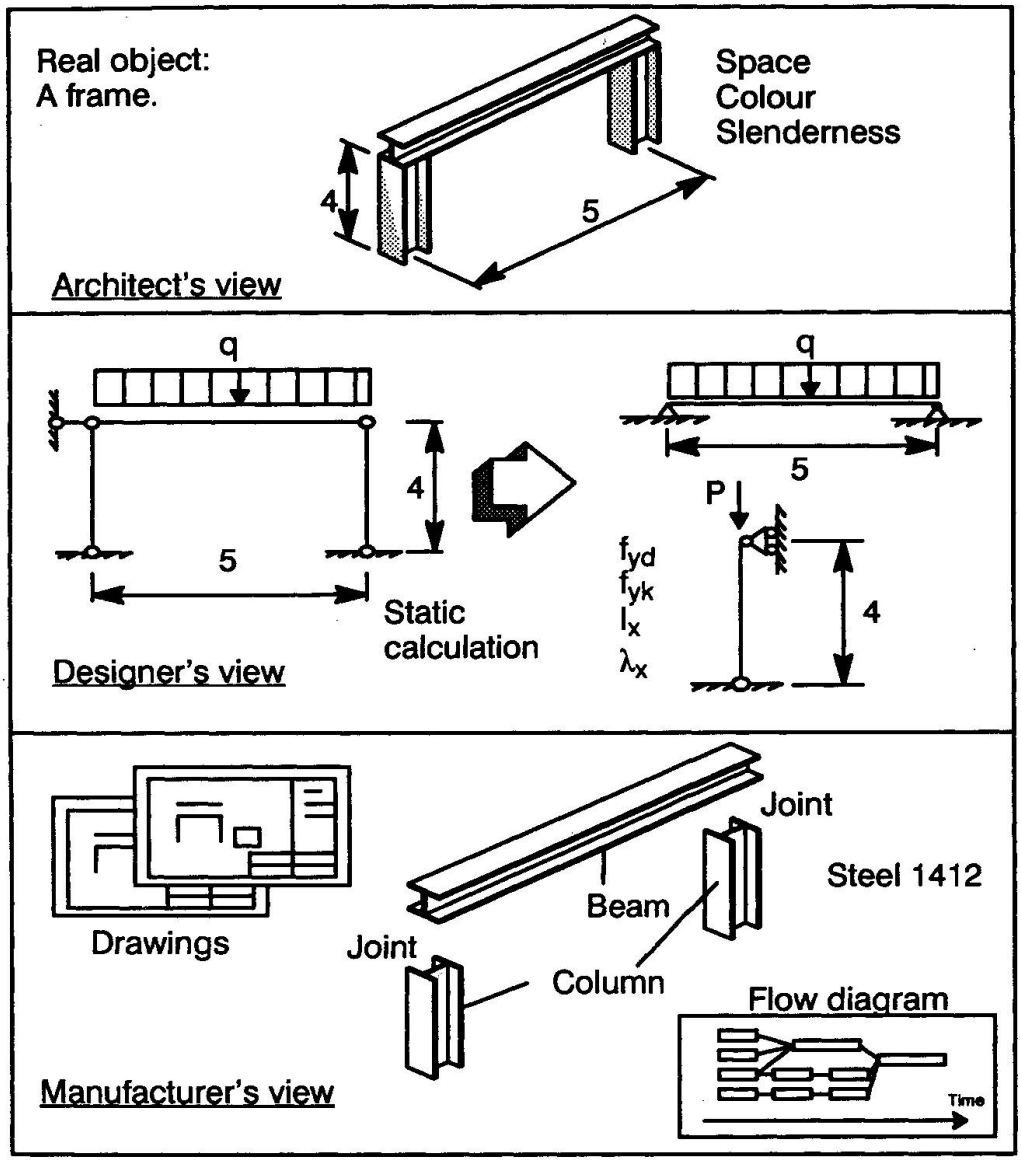


Figure 1. Different abstract representations used by different actors in the process of structural design and manufacturing.

The next generation of computer programs used for structural design must be able to exchange information between each other. The information to be exchanged is, except for geometrical data, such information as object properties, relations between objects, main function of objects etc. The latter are usually referred to as non–geometrical properties. These programs must also have access to knowledge such as experience, heuristics, codes etc. Heuristic knowledge, codes etc are often country or company dependent and are likely to change over time. It is therefore necessary to exclude this type of information from the computer program and to store it in a neutral replaceable format. This neutral storage format can also act as the medium by which different computer programs exchange information. The possibility to exchange information between computer programs will, as mentioned earlier, be crucial in future computer aided design. The medium which is used for the exchange of information and to store heuristic knowledge can be a shared database in which *product models* of the artifact, which is designed, are defined. A product model is a computerized model of a product component. If the shared database is coupled to a production system (expert system) it will be possible to include all necessary knowledge

described above (heuristics, codes, non–geometrical information etc). Furthermore, the production system can include knowledge about the computer environment and therefore acts as an intelligent interface between the database and the different computer programs (figure 2).

The information stored in the database will consist of two parts. One part of the database will contain design rules, frames for product models, codes etc. This information is not likely to change during the design process. The second part of the database will contain unique information for each project. The information should be stored in such a way that all aspects and needs that different actors have during the design process can be utilized. In the the further discussion we will refer to the coupled system consisting of the shared database and the production system as the *intelligent database*.

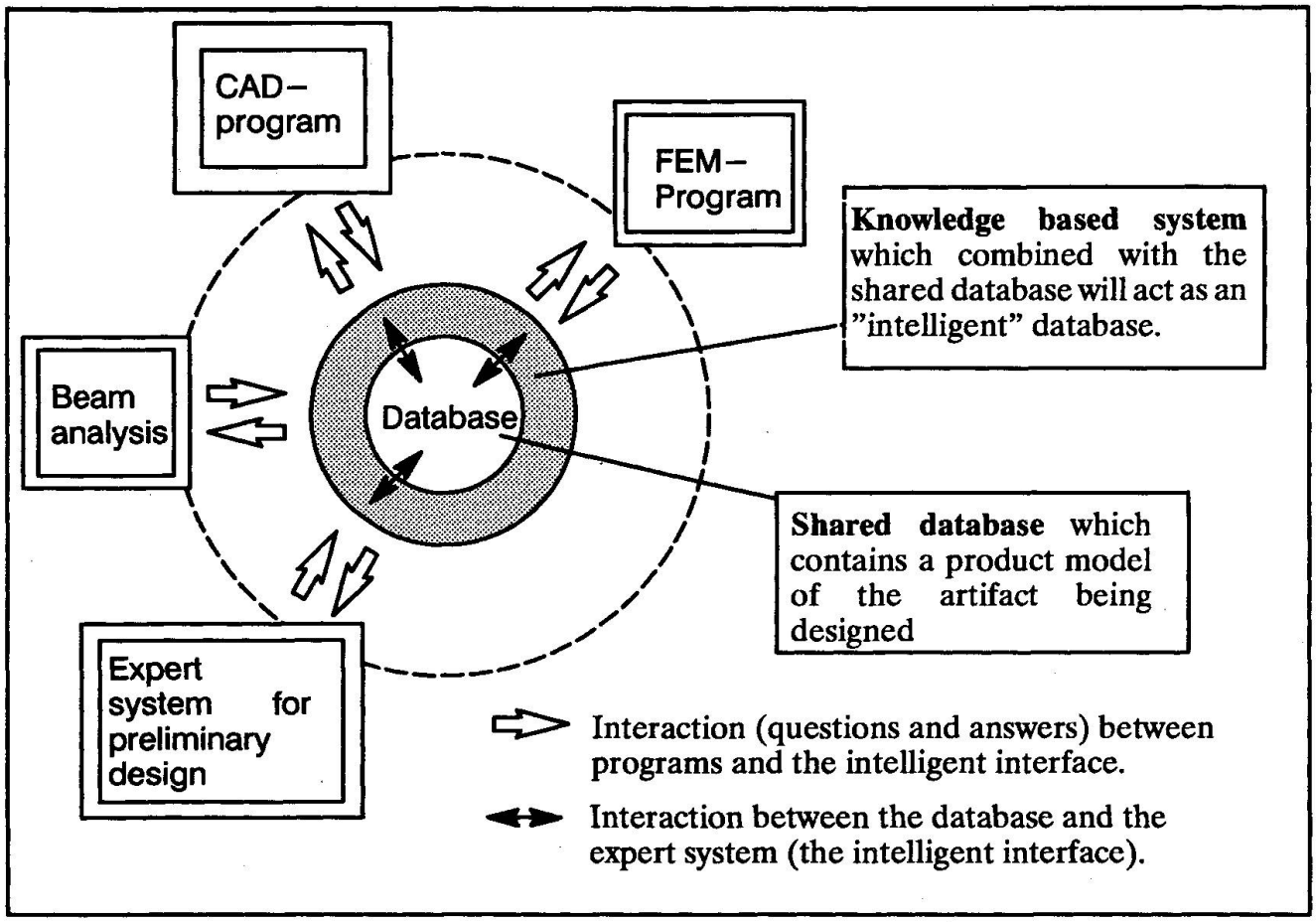


Figure 2. Expert system which will act as an intelligent database, e.g. when exchanging information between different computer programs.

The idea in this discussion is that the intelligent database should include product models which fully describe the artifact to be designed. Different computer programs should not be allowed to function as isolated units. All information should be stored in the shared database. The production system which includes heuristic rules, codes etc will check for any inconsistency in a proposed solution.

2.2 Product models

A product model is a computerized model of a product component. The product model should contain all necessary information about the product. This information includes geometrical shape, how the product should be assembled, tolerances, environmental dependencies etc. An ideal product model

should also include a *history mechanism* which allows us to store information about different design steps. This will enable us to see why and how an artifact has been designed. The use of an object–oriented approach to build product models is perhaps the most promising method. In Sweden some work has been done to develop methods for classifying building details for CAD–systems (Lundberg et al., 1989). This work can among others be regarded as a first step towards the use of product models.

It is necessary that the product model we use can reflect all aspects of building design. Different actors in the design process have different needs. The product model must be able to describe the overall structure (house, bridge etc) as well as the structural subcomponents (frames, columns, doors etc). Each subcomponent as well as the assembled subcomponents will be described as a separate product model. The product model must — apart from describing the component itself — also be able to include information about the relations that the component has *both* to other components *and* to the main structure. It is therefore, when defining product models, necessary to use a technique that allows us to build hierarchical datastructures.

This hierarchical structure can easily be modelled by using an object–oriented approach. Objects are, in object–oriented systems, usually defined by using datastructures called *frames*. A frame is a datastructure which contains all information about a concept. A concept can be a general structural concept such as window, wall, house etc or a specific part in a building such as column–A3, window–B4 etc. Figure 3 shows a frame structure which defines the object "WINDOW–B4". Each object is defined by using *slots* and *facets*. The slots define properties of the object. Each facet defines the value of a slot property, how it can be retrieved and/or restrictions on the value. The frame–based model to store information is commonly used in knowledge–based programming and is fully described in the literature within the field, see for example Walters and Nielsen (1988), Coyne et al. (1989).

OBJECT: WINDOW–B4	
SLOTS:	**FACETS:**
IS–A	(VALUE WINDOW)
PART–OF	(VALUE WALL–1) (RESTRICTIONS (VALUE–IS–A OUTER–WALL))
TYPE	(VALUE M2–STD) (FETCH QUERY)
HEIGHT–OVER–FLOOR	(FETCH DEFAULT)
VERTICAL–PLACEMENT	(FETCH SNAP–TO–MODULAR–SYSTEM)
...	...

Figure 3. Product model definition using a frame structure.

The product model which defines an object must include all known information as well as information about how to retrieve unknown data, restrictions on a value, etc. We must also be able to link known data to its presumptions. A slot value must be recalculated or marked as "unknown" if the presumption to that value is changed. A product model which is used to describe a beam will typically have slots and facets which include information about which codes that are relevant, which program that can be used for structural analysis etc.

3. PREBRI

Preliminary design is a task which needs heuristics and experienced–based knowledge to be carried out. Within the field of preliminary bridge design this means knowledge about costs, material, strength, structural design etc. PREBRI is a prototype expert system, implemented by the ART programming tool, that shows how this type of knowledge can be structured and implemented in computer programs.

3.1. Application of ART (Automated Reasoning Tool)

The Automated Reasoning Tool (ART) is a programming tool for building expert systems. The ART version used in this project (version 3.0) is Lisp–based and runs on different UNIX workstations and mainframes. Here an Apollo workstation DN3500 with 16 MBytes of primary memory (RAM) and 380 MBytes hard disk was used. ART includes facilities such as, rule–based programming, frame–based knowledge representation and object–oriented programming. ART also includes a method for modelling hypothetical alternatives or/and situations that change dynamically over time ("viewpoint mechanism"). For detailed information we refer to the ART manuals [2][3].

3.2. Internal structure of PREBRI

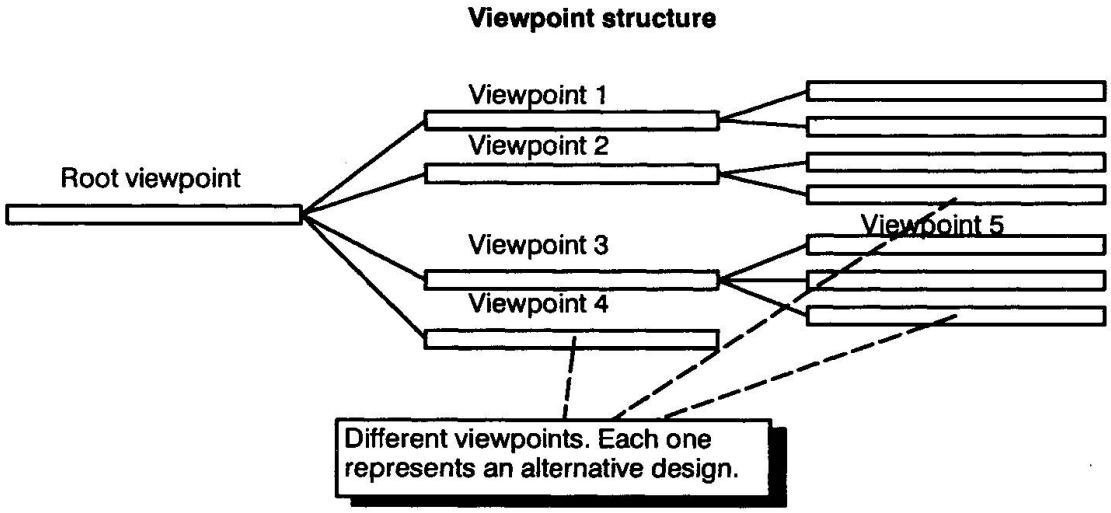
An expert system for preliminary design requires heuristics and experience–based knowledge to give a useful solution. Within the field of preliminary structural design this includes knowledge about costs, materials, maintenance requirements etc. In all such programs it is very important that the user has full insight and control over the system. The expert system should be a support to the user in the design process and not just deliver more or less ready–made solutions.
An expert system for preliminary design should include the following parts.

1. **User interface,** for interaction between the user and the program. The user interface should be developed and specialized for the actual problem under consideration. It must be able to represent knowledge in an effective and natural way.
2. **Domain description,** which includes knowledge such as codes, objects, relations between component parts etc.
3. **Process rules (for design),** which are rules that describe the solution process. These rules include knowledge about the actual design process within the problem field.
4. **Controlling rules,** which include knowledge for detecting errors, checking codes etc. These rules act in the background.
5. **Interface to external programs.** This is a very important part of many expert systems. It is important that a program can interact with other programs or databases in an easy way. This requires a standard for the exchange of information.
6. **Inference engine,** which uses rules and knowledge from points 2, 3, and 4 above to retract or assert information in the internal knowledge base.

The prototype PREBRI was developed in order to investigate how the parts defined by points 2–4 above could be suitable described and implemented in an expert system for preliminary structural design in general and for preliminary design of bridges for motorway crossings in particular. The knowledge in the prototype also include demands on the design such as structural and constructional constraints, costs etc. A simple user interface has also been developed to be used for testing of the prototype.

The domain knowledge (point 2) contains a general description of the objects in the problem domain of the design of a *bridge crossing a motorway.* A lot of effort has been spent in order to make a useful description of the bridge components and of their interrelations. These descriptions can be seen as a first step to a product model for short span bridges. The design rules (point 3) are rules which describe the design process for preliminary bridge design. These rules are the "engine" of the expert system. They propose new solutions, find and retract impossible solutions, recommend the next step in the design process etc. The controlling rules are of "general design" character and concern for instance overall geometry, beam depth etc.

The main idea with the solution process in PREBRI is that all types of solutions which are possible so far are described in different parts of the database (viewpoints). The information stored in each of these parts of the database is true under one or several assumptions (hypotheses) made by the program or by the user. This means that the program uses three kinds of facts, *true facts, assumed facts* and *consequential facts. True facts* are facts that are globally true. *Assumed facts* are hypotheses made by the user or by the program. The assumptions offer a way to describe possible alternative solutions and they form an important part of PREBRI. *Consequential facts* are facts which are true under a certain assumption (figure 4).



Contents of viewpoint 3
Assumption: (Bridge–type CONCRETE_FRAME)
Consequential true facts under the assumption above: (Max–length–of–bridge 70) (Min–length–of–bridge 25) Facts from the root viewpoint.

Contents of viewpoint 5
Assumption: (Number–of–spans 2)
Consequential true facts under the assumption above: (Bridge–type CONCRETE_FRAME) (Max–length–of–bridge 70) (Min–length–of–bridge 30) (Type–of–support–1 RIGID) (Type–of–support–2 RIGID) (Type–of–mid–support FREE) (Min–length–span 15) (Max–length–span 32)

Figure 4. Viewpoint structure.

The whole database (the set of viewpoints) that simultaneously describes different possible solutions forms a tree–structure. The *root–node (root viewpoint)* of that tree–structure is the "global database" which includes facts that are always true independently of all part–solutions chosen. This *root viewpoint* thus includes facts such as "site properties", "minimal bridge length", "minimal bridge width" etc.

From the *root viewpoint* we can make assumptions concerning different possible bridge designs. Each of these assumptions will create a new part of the database (viewpoint). From each viewpoint we can make new assumptions which create new viewpoints etc.

The advantages with this approach is that the user can choose which viewpoint (part of the database) to evolve further. This will probably be that viewpoint which in his opinion is the most promising one. The user will therefore have full control over the system. Assumptions can be made by the user or by the program. The program will make assumptions if it will find a small number of different approaches to continue from a specified point in the solution process. The user can choose to make assumptions if he wants to test an idea or if he is not sure which solution (among several) that is the best. If the user inserts new information in a viewpoint which makes child viewpoints illegal, then all the illegal viewpoints will disappear from the viewpoint structure and all its contents will be deleted.

3.3. Knowledge acquisition

An important part and perhaps the most difficult one when building expert systems is knowledge acquisition. Knowledge acquisition is widely described and discussed in the literature. See among others Harmon and King (1985), Hart (1986), Walters and Nielsen (1988), Wolfgram et al. (1987).

In structural design we have at least three sources from which we can retrieve knowledge. They are literature, domain experts and examples. In PREBRI we use all three forms of sources. The expert knowledge was received from three experts within the field of bridge design (having 23, 30 and more than 40 years of bridge design experience respectively).

The experts pointed out the importance of reusing previous solutions. Even though all bridges are individual it is likely that some bridges will have a lot in common. One way to reuse information is to build a database, which includes all relevant information about each bridge. The information stored for each bridge should be a description containing both background information and the final solution. When we wish to design a new bridge we may then match the information about that bridge with the background information in the database. In this way we can reuse solutions already made and make necessary changes in these. This method of using examples can be an alternative or complement to the method of using rules. In PREBRI we don't use such a database with examples. Instead we have used examples to write rules.

3.4. Knowledge representation

The domain knowledge in PREBRI is represented using an object–oriented approach combined with rules. The type of objects used in PREBRI are *classes*, *entities* (description of a "real object") and *relations*. In the present version of PREBRI there are about 450 objects and 10 types of relations defined. The *classes* are used to describe different types of bridges. The types with their subclasses are described using a so called class–tree which defines the hierarchical structure of the classification. Each node in the tree represents a specific class of bridges and includes information (attributes) specific for that class. The root node is the most general class. In PREBRI the most general class is the class "bridge". The leaf of the class–tree is the most specific description of a bridge type. Each subclass inherits information from its superclass.

The structural components of a bridge are also defined as objects. These objects are for example beams, supports, columns etc. The *entity* objects are descriptions of real world objects such as column, support etc. Several entities can be combined to form a new entity. Each entity is connected to its environment using relations such as *part–of*, *below*, *supported–by* etc. The object description of an entity contains information concerning location, size, function, connection to other entities etc. *Relations* are definitions of connections or relations between objects or classes.

The rules in PREBRI are of *if–then* type. The rules are of two kinds: *design–process rules* and *control rules*. An example of a design–process rule can be a rule for estimating the construction depth of the main structure of a bridge (figure 5). An example of a control rule is a rule which checks the system for inconsistency.

```
(defrule estimate-min-construction-depth-slab-bridge
  (schema ?name
    (instance-of slab-bridge)
    (type-of-bars normal)
    (number-of-span 1)
    (min-length-span ?min-length))
  =>
    (bind ?min-c-h (* 0.055 ?min-length))
    (modify (schema ?name (min-construction-depth ?min-c-h))
```

Figure 5. A typical rule in PREBRI.

3.5. Graphical interface

The aim of this project has not been to develop an efficient and sophisticated interface between the user

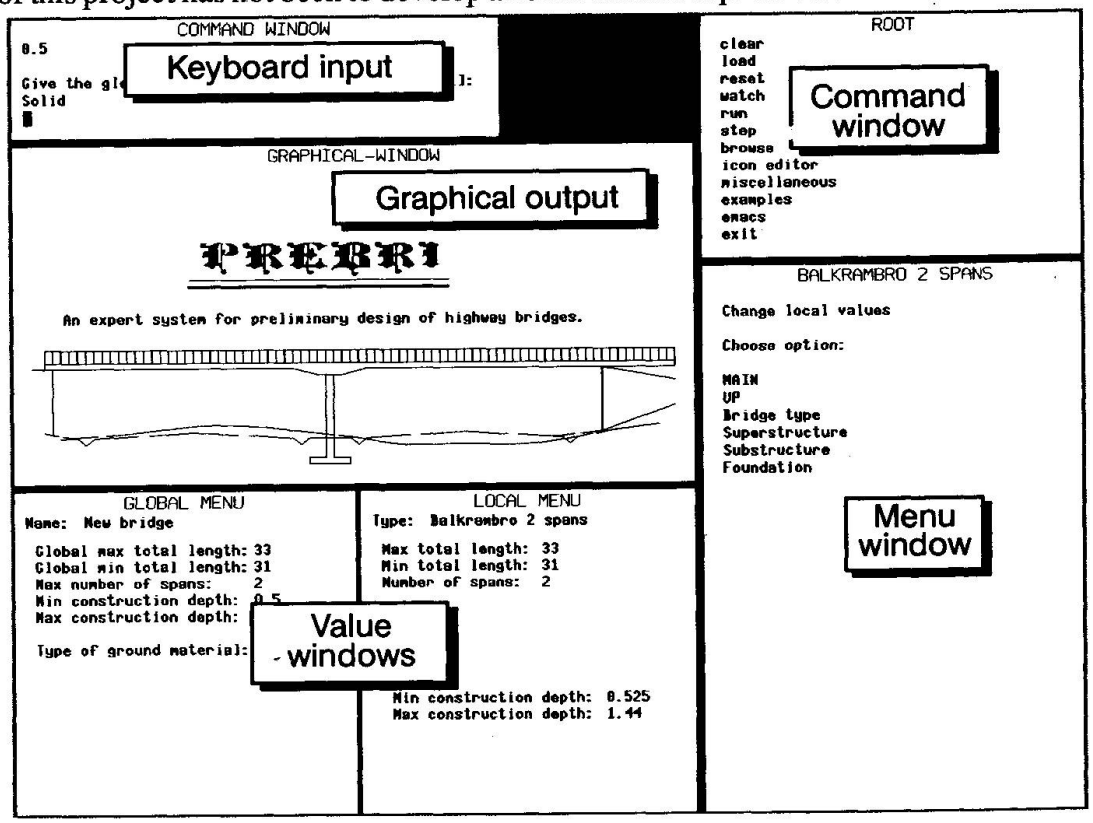


Figure 6. Graphical interface of PREBRI.

and the program. The building of such an interface is a complicated task which includes aspects such as visuality, clearness, usability etc. In order to accomplish a good interface we need better understanding of the design process, a mental model of the user etc. The interface of PREBRI is therefore rather simple and contains 5 windows, cf. figure 6. A keyboard input window, a graphical window to visualize the current bridge type, two value windows to show current values (length, bridge type, active hypothesis etc) and two menu windows, the ART programming tool command window and the PREBRI menu

window. The command window can be used to inspect the viewpoint tree, browse facts and relations etc. The PREBRI menu window allows the user to interact with the program. The PREBRI menu system has been developed using object–oriented methods.

4. REFERENCES

1. AHLENIUS, P. (Ed.), Broprojektering – En Handbok (Preliminary Bridge Design – A Handbook. In Swedish), Vägverket, Borlänge 1988.
2. ART, Reference Guide, Volume 1–2, Inference Corporation, Los Angeles 1987.
3. ART, Programming Tutorial, Volume 1–4, Inference Corporation, Los Angeles 1987.
4. BRONORM 88, (Bridge Code 88. In Swedish), Vägverket, Borlänge 1988.
5. BROTYPER Tabeller, (Bridge types. Tables, In Swedish), Vägverket, Borlänge 1988.
6. COYNE, R. D., ROSENMAN, M. A., RADFORD, A. D., BALACHANDRAN, M., GERO, J. S., Knowledge Based Design Systems, Addison–Wesley, New York 1989.
7. GARRETT, J. H. Jr., Object–oriented representation of design standards, IABSE Colloquium "Expert Systems in Civil Engineering", Bergamo, October 1989.
8. HARMON, P., KING, D., Expert Systems, John Wiley & Sons, New York 1985.
9. HART, A., Knowledge acquisition for expert systems, Kogan Page, London 1986.
10. KATZ, T,. Engineering Design, Manufacture and Test, In (Winstanley, G., 1991), pp 151–193.
11. LUNDBERG, K., LUNDEQUIST, J., LOTZ–MATTSON, M., Metoder för klassifikation av bygg– eller mätdelar för CAD–system (In Swedish). Rapport R57:1989. Statens råd för byggnadsforskning, Stockholm 1989.
12. LÖFQVIST, P., Expertsystem för broprojektering (Expert system for bridge planning. In Swedish), Chalmers University of Technology, Division of Steel and Timber Structures, Int. skr. S 90:6, Göteborg 1990.
13. LÖFQVIST, P., Expert system for bridge design, Chalmers University of Technology, Division of Steel and Timber Structures, Int. skr. S 91:4, Göteborg 1992.
14. NISHIDO, T., MAEDA, K., NOMURA, K., Practical System for Type Selection of Bridge Crossing River, IABSE Colloquium "Expert Systems in Civil Engineering", Bergamo, Italy, 1989, pp 311–319.
15. WALTERS, J. R., NIELSEN, N. R., Crafting Knowledge Based Systems, John Wiley & Sons, New York 1988.
16. WINSTANLEY, G. (Ed.), Artificial Intelligence in Engineering, John Wiley & Sons, Chichester, 1991.
17. WINSTANLEY, G., The Engineering Domain and Systems, In (Winstanley, G., 1991), pp 1–32.