Expert Systems, Vol. 21, No. 4, 2004, pp. 183-191

AN EXPERT SYSTEM ON DESIGN OF LIQUID RETAINING STRUCTURES WITH BLACKBOARD ARCHITECTURE

K.W. Chau
Department of Civil & Structural Engineering, Hong Kong Polytechnic University, Hunghom, Kowloon, Hong Kong

F. Albermani
Department of Civil Engineering, University of Queensland, Australia

ABSTRACT: The design of liquid retaining structures involves many decisions to be made by the designer based on rules of thumb, heuristics, judgment, code of practice and previous experience. Structural design problems are often ill structured and there is a need to develop programming environments that can incorporate engineering judgment along with algorithmic tools. Recent developments in artificial intelligence have made it possible to develop an expert system that can provide expert advice to the user in selection of design criteria and design parameters. This paper introduces the development of an expert system in design of liquid retaining structures using blackboard architecture. An expert system shell, Visual Rule Studio, is employed to facilitate the development of this prototype system. It is a coupled system combining symbolic processing with the traditional numerical processing. The expert system developed is based on British Standards Code of Practice BS8007. Explanations are made to assist inexperienced designers or civil engineering students to learn how to design liquid retaining structures effectively and sustainably in their design practices. The use of this expert system in disseminating heuristic knowledge and experience to practitioners and engineering students is discussed.

INTRODUCTION

Design of liquid retaining structures involves many decisions to be made by the designer based on rules of thumb, heuristics, judgment, code of practice and previous experience. Various design parameters to be chosen include configuration, material, loading, etc. A novice engineer may face many difficulties in the design process.

Advances in computer hardware, computer software and engineering methodologies in the recent decades have led to an increased use of computers by engineers. Until recently, the main use of computers by engineers was mainly confined to the number crunching of large volumes of numerical data. In the realm of structural design, this use has been limited almost exclusively to algorithmic solutions (Chau & Lee 1991). Structural design problems are often ill structured. Hence, there is a need to develop programming environments that can incorporate engineering judgment along with algorithmic tools. (Kitzmiller & Kowalik 1987)

In the past decade, the potential of artificial intelligence (AI) techniques for providing assistance in the solution of engineering problems has been recognized. Expert systems are considered suitable for solving problems that demand considerable expertise, judgment or rules of thumb. As a result of years of research in AI, expert systems have emerged as a most promising application covering a wide range of applications (Adeli & Hawkins 1991, Chau 1992, 2004a&b, Chau & Anson 2002, Chau & Chen 2001, Chau et al. 2002, Chau & Cheung 2004, Chau & Ng 1996, Chau & Yang 1994, Chau & Zhang 1995, Kumar 1995, Lin &

Albermani 1998, Sriram 1987). Expert systems have developed into practical problem solving tools that can reach a level of performance comparable to that of a human expert in some specific problem domains. All these applications can be broadly classified into the following categories: diagnosis; design; data interpretation; planning; and education. Areas of early applications of expert systems technology include medical diagnosis, mineral exploration and chemical spectroscopy. Recent developments in artificial intelligence have made it possible to develop an expert system that can provide expert advice to the user in selection of design criteria and design parameters. (Dym & Levitt 1991)

This paper introduces the development of an expert system in design of liquid retaining structures using blackboard architecture with hybrid knowledge representation techniques including production rule system and object-oriented approach. An expert system shell, Visual Rule Studio, is employed to facilitate the development of this prototype system. It is a coupled system in which AI-based symbolic processing is combined with the traditional numerical processing. The expert system developed is based on British Standards Code of Practice BS8007: 1987: Design of concrete structures for retaining aqueous liquids (British Standards Institution 1987). Explanations are made to assist inexperienced designers or civil engineering students to learn how to design liquid retaining structures effectively and sustainably in their design practices. The use of this intelligent tutoring system in disseminating heuristic knowledge and experience to practitioners and engineering students is discussed.

KNOWLEDGE ON DESIGN OF LIQUID RETAINING STRUCTURES

Liquid retaining structure is a structure which is designed and constructed to retain aqueous liquid. It is subject to lateral water pressure and earth pressure when it is located underground. Most of these structures are constructed by reinforced concrete with design life of 50 years in Hong Kong. Crack widths are to be checked to ensure impermeability of concrete and prevention from corrosion of reinforcement.

Normally, two kinds of classification are used regarding liquid retaining structures, i.e. according to the shape or the location. Based on the shape, it is classified as rectangular, circular or polygon. Based on its location, it is classified as underground or above ground.

Compared with circular tank structure with the same width, rectangular liquid retaining structure has larger volume. However, because of stress concentration at corners, rectangular structures will be more vulnerable to failure. It also has a weaker deflection control. With a circular tank design, not all the spaces are utilized. Since a circular structure can be constructed monolithically without any construction joints, it has better strength quality. With precise structural analysis, circular structure has a better control in deflection, crack width, bending moment resistance, axial compression resistance, and shear resistance than rectangular structure. Polygon liquid retaining structure is usually used for aesthetic purposes, such as a fountain in a garden and the retaining height is usually not very high. Its major design consideration is on crack width to ensure its impermeability and is seldom used in industrial or domestic fields.

Underground liquid retaining structure usually has larger base area which cannot easily be supported by structure above the ground. In some cases, the tank is connected to underground pipe network system, in order to reduce maintenance cost, it will be constructed underground to suit the invert level of the pipe network system. The underground structure is mainly subjected to lateral earth pressure or lateral water pressure which is due to the underground water table. Besides checking structural failure mode, bearing capacity of soil and settlement of structure also need to be checked. If the soil bearing capacity does not satisfy the requirement, pile foundation or raft foundation will be required. Liquid retaining structure above the ground is only subject to liquid pressure due to its own retaining liquid. The structure can either rest on ground concrete slab immediately or rest on supports which can be 3-D steel truss, mass concrete blocks, or concrete beam.

There are a great variety of factors affecting the decision in selecting design criteria and design method. These factors are: dimensions, location, ground condition, support condition, groundwater conditions, aesthetic properties, design life, exposure condition, usage, roofing, availability of construction materials.

Liquid retaining structures, like any other engineering structures, should not fail to satisfy any of its performance criteria. According to the Code of Practice BS8007, the two main classes of limit state, which should be considered, are ultimate limit state and serviceability limit state. Ultimate limit state is design against structural failure, including bending moment check and shear force check. Serviceability limit state is design against deflection and crack width. Normal crack width control is 0.2mm while, for severe cases, allowable crack width is 0.1mm. For underground liquid retaining structures, serviceability limit state design is used in checking of bearing capacity of soil. Factors of safety are involved in the design in order to increase the reliability that the structures will perform satisfactorily.

EXPERT SYSTEM

Expert systems are interactive computer programs that incorporate expertise and provide advice on a wide range of tasks. They solve a specific complex problem using reasoning processes that resemble those of human experts, when they solve the same problem. They tend to mimic the decision making and reasoning processes of human experts by providing expert advice, answering questions, and justifying their conclusions. An expert system can contain the purpose of teaching. Only explanations are not sufficient in the system. The problem should be dealt with by communication between the user and the system. In other words, the system has to engage the user in a dialogue actively and systematically.

Figure 1 shows the schematic view of an expert system. These systems typically consist of the following three basic components: knowledge base, context and inference mechanism. The heart and core of any expert system is the knowledge base, which is usually a collection of rules, typically in the form of IF..THEN….. The knowledge base is a collection of general facts, rules of thumb and causal models of the behavior of the problem domain. It contains the knowledge specific to the domain of the problem to be solved. Other forms of representations commonly used are logic, frame-based schemes, nets and, more recently, the object-oriented approach. For expert system developers, rule-based system tends to be more easily understood and thus accepted. However, because of its modularity, data abstraction and inheritance characteristics, object-oriented programming will likely subsume other approaches in the very near future.

The context is a workspace for the problem constructed by the inference mechanism from the information provided by the user and the knowledge base. It contains facts that reflect the current state of the problem. The organization of the context depends on the nature of the problem domain. The context builds up dynamically as a particular problem is being considered. The context is used by the inference mechanism to guide the decision making process.

The inference mechanism monitors the execution of the program by using the knowledge base to modify the context. It manipulates the context using the knowledge base. A number of problem solving strategies exist in current expert systems; such as forward chaining, where the system works from an initial state of known facts to a goal state.

In addition to the three main modules described above, the system should also be provided with three other components that are not necessarily part of every expert system but are desirable in a final product: a graceful user interface, an explanation facility and a knowledge acquisition module.

The function of the user interface module is to accept a problem description from the user and to interact with the rest of the system in order to analyze the problem or augment the capability of the system. It provides an interface between the user and the expert system, usually as a command language for directing execution. The interface is responsible for translating the input as specified by the user to the form used by the expert system and for handling the interaction between the user and the expert system during the decision making process.

The explanation module provides explanations of the inferences used by the expert system. This explanation can be a priori – why a certain fact is requested, or a posteriori – how a conclusion was reached.

The knowledge acquisition module serves as an interface between the experts and the expert system. It provides a means for entering domain specific knowledge into the knowledge base and revising this knowledge when necessary.

COMPARISON WITH CONVENTIONAL PROGRAMMING

Conventional programs consist of a set of statements whose order of execution is predetermined. These programs are very inflexible; updates need considerable effort, because the programmer has to locate the appropriate place to update in the predefined sequence. The programmer must ensure completeness, i.e., that the program performs correctly for all possible combination of conditions, and uniqueness of the solution, i.e., that the output is unique for every possible input. The user perceives the program as a blackbox, where the program outputs results for the input provided; he does not have any idea as to why the program has produced certain results.

An expert system eliminates some of the impediments posed by conventional programs by making a clear distinction between the knowledge base and the control strategy. This partitioning allows for incremental addition of knowledge, without manipulating the overall program structure; the programmer need not guarantee completeness. Further, by ranking several alternatives either by an evaluation scheme or by the use of inexact inference methods, several solutions can be provided for a particular set of input conditions, thus relaxing the uniqueness constraint. The user can also question the results produced by the program through the explanation module. It is clear that an expert system offers a powerful programming environment for the development of engineering software, since engineering involves extensive use of heuristics.

BLACKBOARD ARCHITECTURE

The blackboard architecture is intended to support development of systems in domains characterized by interaction between diverse sources of knowledge and hence provides a framework for integrating knowledge from several sources. The blackboard serves as a global data structure which facilitates this interaction. A common analogy may be made to problem-solving in domains where a number of experts in different areas of specialities co-operate over the solution which any one of them could never achieve alone. In order to facilitate this process, they agree to use a blackboard to post (or write) any partial result they can contribute separately. Each expert takes turns to write on the blackboard and, in case more experts wish to write simultaneously, the conflict is resolved by some pre-defined strategy.

The blackboard architecture has been successfully used in solving a wide range of tasks, such as speech recognition, signal processing, and planning. (Engelmore & Morgan 1988) A blackboard system consists of a number of knowledge sources that communicate through a blackboard and are controlled by an inference mechanism. Figure 2 shows the architecture of a blackboard system. The main components of a typical blackboard system are entries, knowledge sources, blackboard, and inference mechanism.

Entries are the immediate results produced by the system. In a typical system, each entry has a certainty factor as well as a specification.

The knowledge base consists of a number of knowledge sources (KSs), which contain the knowledge. These knowledge sources are independent chunks of knowledge and do not directly communicate with each other. Instead, they participate in the problem solving process by creating entries in a global database – the blackboard. Knowledge modules look at the blackboard to see if suitable data is present to trigger their execution. If they are selected, execution results in new or altered data on the blackboard, which will then trigger other knowledge modules. Solving a problem using a blackboard architecture is based on cooperation of the knowledge modules present. Each knowledge source consists of a condition-action pair. Actions are executed whenever the conditions are satisfied in the blackboard.

The blackboard or context consists of the information or entries generated by the knowledge sources during the problem solving process. It is organized into a number of levels each representing different aspects or stages of the solution process. These levels depict an *a priori* plan for the solution of a problem that can be naturally decomposed into a set of levels. Each level contains objects and attributes that are important to the representation of the problem. Normally, knowledge sources are specific to certain levels in the blackboard, i.e., the activation of a certain knowledge sources depends on the entries generated at certain levels in the blackboard, while the actions of the knowledge source modify entries at some other level. The main units in the blackboard are hypotheses. The hypotheses are either primary guesses about particular aspects of the problem or partial solutions. Hypotheses at various levels are related through structural relationships.

The inference mechanism consists of two main components: the agenda or scheduler, and the monitor. The agenda keeps track of all the events in the blackboard and calculates the priority of execution for knowledge sources that were generated as a result of the activation of other knowledge sources. It is a list of knowledge sources or rules to be executed in the next cycle.

Based on the success or failure of a particular rule, new rules may get added on to it or some may be deleted from it. The basis of giving priorities to the rules on the agenda may vary from system to system. The monitor takes the element with the highest priority and executes it. Several problem-solving strategies can be implemented using the monitor.

Because of its modularity, the blackboard architecture enables easy incremental development of a software system. Developers can integrate different methods of knowledge representation in a single system because of the modularity of knowledge sources.

KNOWLEDGE REPRESENTATION

Broadly speaking, structural analysis is described as a three-stage process involving: modeling, solution and evaluation. Before knowledge can be represented in structural analysis, the type of knowledge involved must be identified and classified. Static knowledge consists of definitions, axioms and laws. These may be a priori or the result of scientific investigation. Dynamic knowledge refers to heuristics, which is related to the process of search or to knowledge based on experience. Dynamic knowledge is not deducible from any axiom, rather it is generally gained from experience. Dynamic knowledge can also be described as ‘shallow’, meaning that it cannot provide us with explanations of why certain decisions should be made. Static knowledge is ‘deep’ in that it is deduced from fundamentals.

There are several approaches for declarative representation of knowledge that are available in the AI literature, the following three, among others, being the major ones: rule-based production system, frames and object-oriented programming. A production system is a collection of rules and is believed to be good at describing heuristic knowledge. A frame system, on the other hand, is suitable for a complex and rich representation of knowledge, such as fundamental principles (categorized as static knowledge). Object-oriented programming concept is used, in which a computer program consists of a number of independent objects that process jobs by exchanging information they need via messages. To apply object-oriented program development, data representations for the model must be specified. There are three steps in the development of an object-oriented program: selection of classes, specification of the classes, implementation of the classes. Here, the word ‘class’ refers to a description of a set of similar objects; one member object in a class is called an ‘instance’. It seems a good idea to utilize both representations to solve structural design problems.

USE OF EXPERT SYSTEM SHELL

To facilitate the development of expert systems, expert system programming environments or shells have been developed. These system shells contain specific representation methods and inference mechanisms. The knowledge base and explanation facility of the system have been developed using a commercially available expert system shell called Visual Rule Studio which is a hybrid application development tool that integrates object-oriented techniques and expert system technology with traditional, procedural programming (RuleMachine Corporation 1998). Visual Rule Studio installs as an integral part of Microsoft Visual Basic 6.0 and appears within Visual Basic as an ActiveX Designer. As a part of the Visual Basic Integrated Development Environment, using a RuleSet in the application is similar to using a form or other Visual basic Designer.

By isolating rules as component objects, separate from objects and application logic, Visual Rule Studio allows developers to leverage the proven productivity of today’s component oriented development tools, such as Visual Basic. With Visual Rule Studio, rule development becomes a natural part of the component architecture development process. The complex and time consuming problems of integrating multiple development tools and managing incompatible object models no longer exist. Visual Rule Studio becomes an integrated part of the Visual Basic development and produces objects that can interact with virtually any modern development product.

Visual Rule Studio objects are used to encapsulate knowledge structure, procedures, and values. An object’s structure is defined by its class and attribute declarations within a RuleSet. Object behavior is tightly bound to attributes in the form of facets, methods, rules, and demons. Figure 3 shows the structure of Visual Rule Studio components. Each attribute of a class has a specific attribute type. The Visual Rule Studio attribute types are compound, multicompound, instance reference, numeric, simple, string, interval, and time. Each attribute can have many facets associated with it. Facets provide control over how the inference engines process and use attributes. Each attribute can also have methods associated with it. Methods establish developer-defined procedures associated with each attribute. The set of backward-chaining rules that conclude the same attribute is called the attribute’s rule group. The set of forward-chaining demons that reference the same attribute in their antecedents is called the attribute’s demon group.

The Visual Rule Studio inference engines control the strategies that determine how, from where, and in what order a knowledge base draws its conclusions. These inference strategies model the reasoning processes an expert uses when solving a problem. Visual Rule Studio supports these types of inferencing strategies: backward chaining, forward chaining and hybrid chaining. Each of these inferencing strategies acts on specific knowledge base components. Backward chaining inferencing starts with a specific hypothesis, or set of hypotheses, called the agenda. In backward chaining, the inference engine works backward from the agenda, pursuing a hypothesis via its search order strategies, which can be composed of the session context, the knowledge base rules, WHEN NEEDED methods, Query Objects, or the default value. Forward chaining inferencing starts with known data or conditions and determines what can be concluded from that data. Forward chaining uses a knowledge base’s demons and WHEN CHANGED methods. Hybrid inferencing is a mixed strategy that combines backward chaining and forward chaining.

EXPERT SYSTEM ON DESIGN OF LIQUID RETAINING STRUCTURES

The system being developed combines expert systems technologies, object-oriented programming, relational database models and hypertext/graphics in a windowing environment. It runs under and follows the conventions of Microsoft Windows. In a windowing system, any types of display windows can be represented as objects, each with its own private data or information. By defining various types of windows as different classes, such as checkbox group, hyperregion, promptbox, pushbutton, textbook, etc., they can inherit common characteristics or/and possess their own special properties.

The knowledge used has been acquired mostly from written documents such as code of practice, textbooks and design manuals and complemented by experienced engineers involved with the design of liquid retaining structures. The domain knowledge is translated into procedures and methods using object-oriented representation. The system can be compiled and encrypted to create a run-only system. This run-only system can be installed on a microcomputer for office use or on a portable laptop to be used in the field. The user can always overrule any design options and recommendations provided by the system. In other words, it plays the role of a knowledgeable assistant only.

The input data provided by the user will be rejected if it is not within the range specified. It can explain its line of reasoning for obtaining an answer. It provides information about any individual member in multi-window graphics text display where graphic images are combined with valuable textual information. This kind of intelligent graphics is extremely valuable to structural designers because it enhances their confidence in the design provided by the expert system.

The system offers a state-of-the-art user interface. The use of a mouse or other pointing device makes the data entry a simple task even for novice computer users. As such, users simply point and click their way through the process to appreciate the dynamic behavior of the system. Input data entry is kept at minimum. Input data are provided by the user mostly through selection of appropriate values of parameters from the menus and providing answers to the queries made by the system. An expert system can help the novice engineers or students to learn more knowledge of the topic through using the system. Explanations are made to assist them to learn how to design liquid retaining structures effectively and sustainably in their design practices. Implementation of the system reduces the dependence on experienced designers for routine design works and frees them to do more innovative design works.

TYPICAL CONSULTATION SESSION

To demonstrate how this expert system assists engineer in the preliminary design of liquid retaining structure, a sample run is demonstrated in this section. A typical example of liquid retaining structure is used to illustrate its application. An rectangular shape liquid retaining structure with a single compartment located underground, having a volume of 100 m^3, a depth of 5 m and breadth/width ratio of 1.2, is designed under a severe exposure environment, i.e. the maximum design crack width is 0.2mm. The purpose of the design is to find a feasible optimum structural section in term of minimum unit cost per meter length, based on the standard slab thickness and reinforcement size as well as bar spacing. Figures 4 to 7 show some typical screen displays during the execution of the expert system. Little explanation facility is required since each screen display was designed to be user-friendly to follow.

CONCLUSIONS

It has been demonstrated that the hybrid knowledge representation approach combining production rule system and object-oriented programming technique to the design of water retaining structures is possible with the implementation of blackboard system architecture. It is appropriate to integrate algorithmic and symbolic programming on structural design into a single computer-aided environment running under a Windows platform. The educational spin-off of expert systems in training novice engineers or in transferring knowledge cannot be overemphasized.

REFERENCES

Adeli, H. and Hawkins, D.W. (1991) A hierarchical expert system for design of floors in

highrise buildings, *Computers & Structures*, 41(4): 773-778.
British Standards Institution (1987) *Design of Concrete Structures for Retaining Aqueous Liquids*, British Standards Institution, London.
Chau, K.W. (1992) An expert system for the design of gravity-type vertical seawalls, *Engineering Applications of Artificial Intelligenc*e, 5(4), 363-367.
Chau, K.W. (2004a) A prototype knowledge-based system on unsteady open channel flow in water resources management, *Water International*, 29(1), 54-60.
Chau, K.W. (2004b) Knowledge-based system on water-resources management in coastal waters, *Journal of the Chartered Institution of Water and Environmental Management*, 18(1), 25-28.
Chau, K.W. and Anson, M. (2002) A knowledge-based system for construction site level facilities layout, *Lecture Notes in Artificial Intelligence*, 2358, 393-402.
Chau, K.W. and Chen, W. (2001) An example of expert system on numerical modelling system in coastal processes, *Advances in Engineering Software*, 32(9), 695-703.
Chau, K.W., Cheng, Chuntian and Li, C.W. (2002) Knowledge management system on flow and water quality modeling, *Expert Systems with Applications*, 22(4), 321-330.
Chau, K.W. and Cheung, C.S. (2004) Knowledge representation on design of storm drainage system, *Lecture Notes in Artificial Intelligence*, 3029, 886-894.
Chau, K.W. and Lee, S.T. (1991) Computer Aided Design Package `RCTANK' for Analysis and Design of Reinforced Concrete Tanks, *Computers and Structures*, 41(4): 789-799.
Chau, K.W. and Ng, V. (1996) A knowledge-based expert system for design of thrust blocks for water pipelines in Hong Kong, *Water Supply Research and Technology - Aqua*, 45(2), 96-99.
Chau, K.W. and Yang, W.W. (1994) Structuring and evaluation of VP-expert based knowledge bases, *Engineering Applications of Artificial Intelligence*, 7(4), 447-454.
Chau, K.W. and Zhang, X.N. (1995) An expert system for flow routing in a river network, *Advances in Engineering Software*, 22(3), 139-146.
Dym, C.L. and Levitt, R.E. (1991) *Knowledge-Based Systems in Engineering*, McGraw-Hill, New York.
Engelmore, R. and Morgan, T. (1988) *Blackboard Systems*, Addison_Wesley, Wokingham.
Kitzmiller, C.T. and Kowalik, J. S. (1987) Coupling Symbolic and Numeric Computing in Knowledge-Based Systems, *AI Magazine*, Summer: 5-90.
Kumar, B. (1995) *Knowledge Processing for Structural Design*, Topics in Engineering Vol. 25, Computational Mechanics, Southampton.
Lin, S. and Albermani, F., 2001, Lattice-dome design using A knowledge-based system approach, *Computer-aided Civil and Infrastructure Engineering*, 16(4), 268-286.
RuleMachines Corporation (1998) Visual Rule Studio Developer's Guide, Indialantic.
Sriram, D. (1987) *Knowledge-Based Approach for Structural Design*, Topics in Engineering Vol. 25, Computational Mechanics, Southampton.

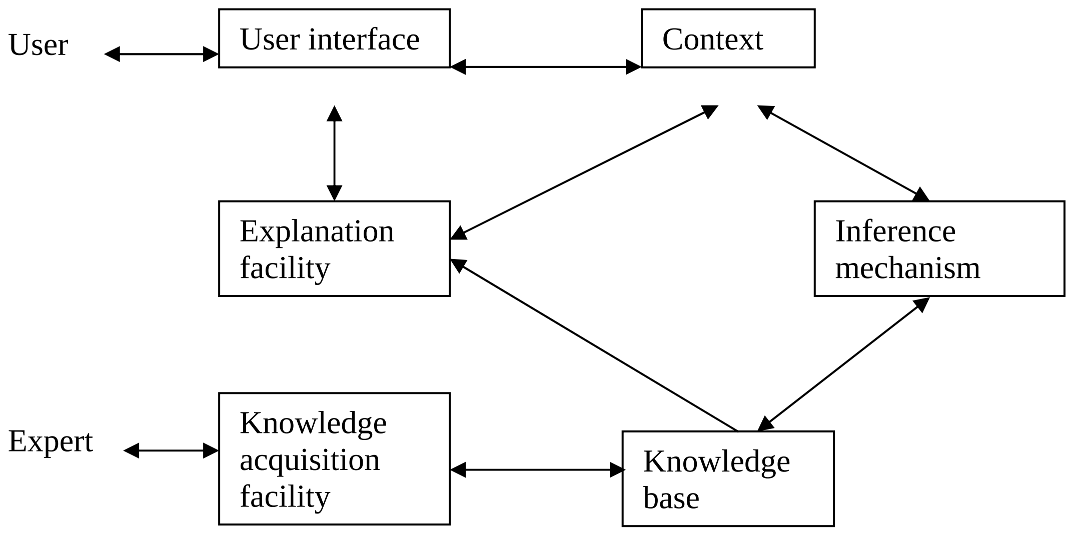


Figure 1. Schematic View of an expert system

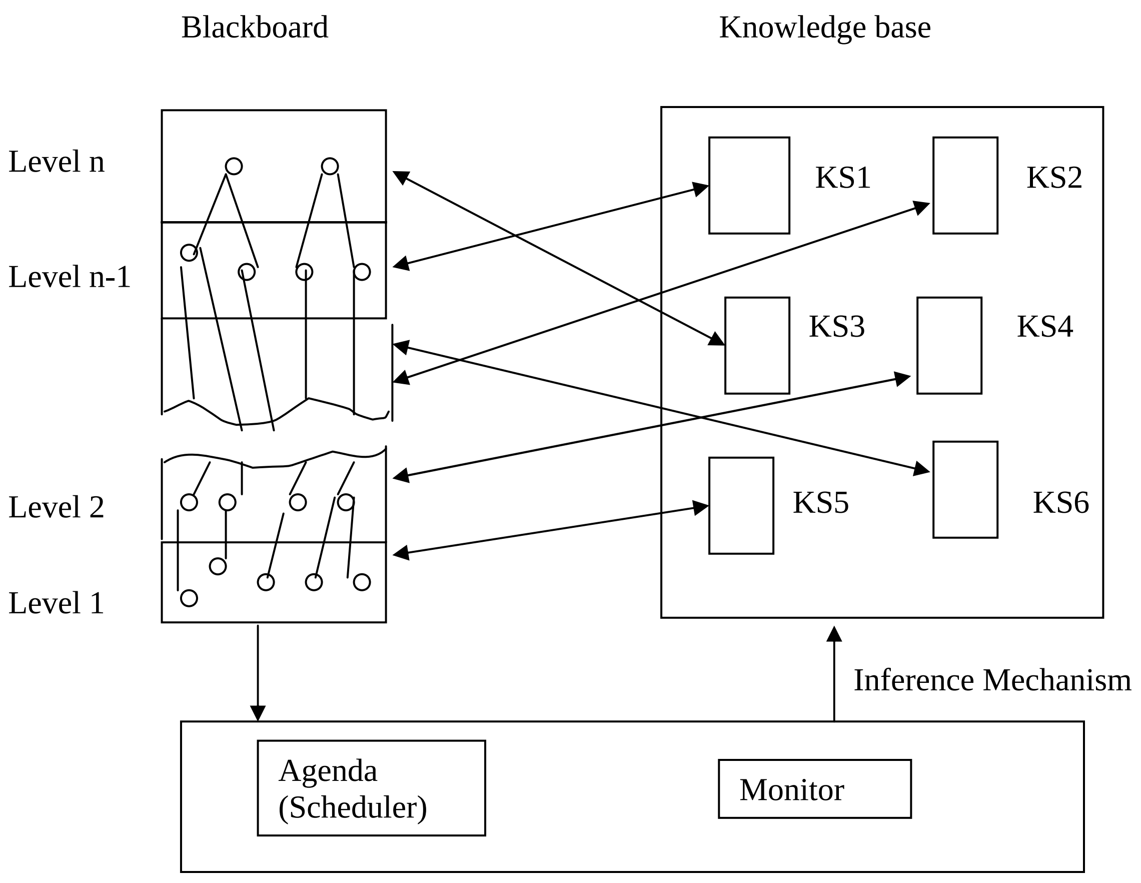


Figure 2. The Blackboard Architecture

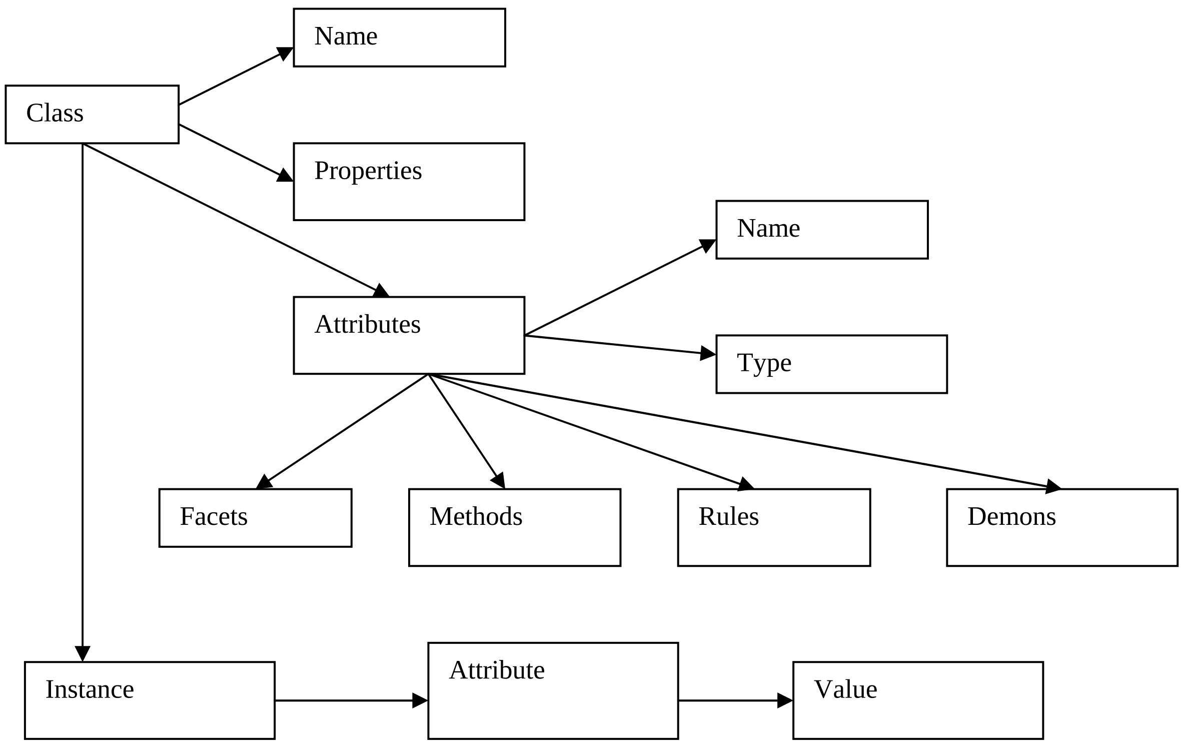
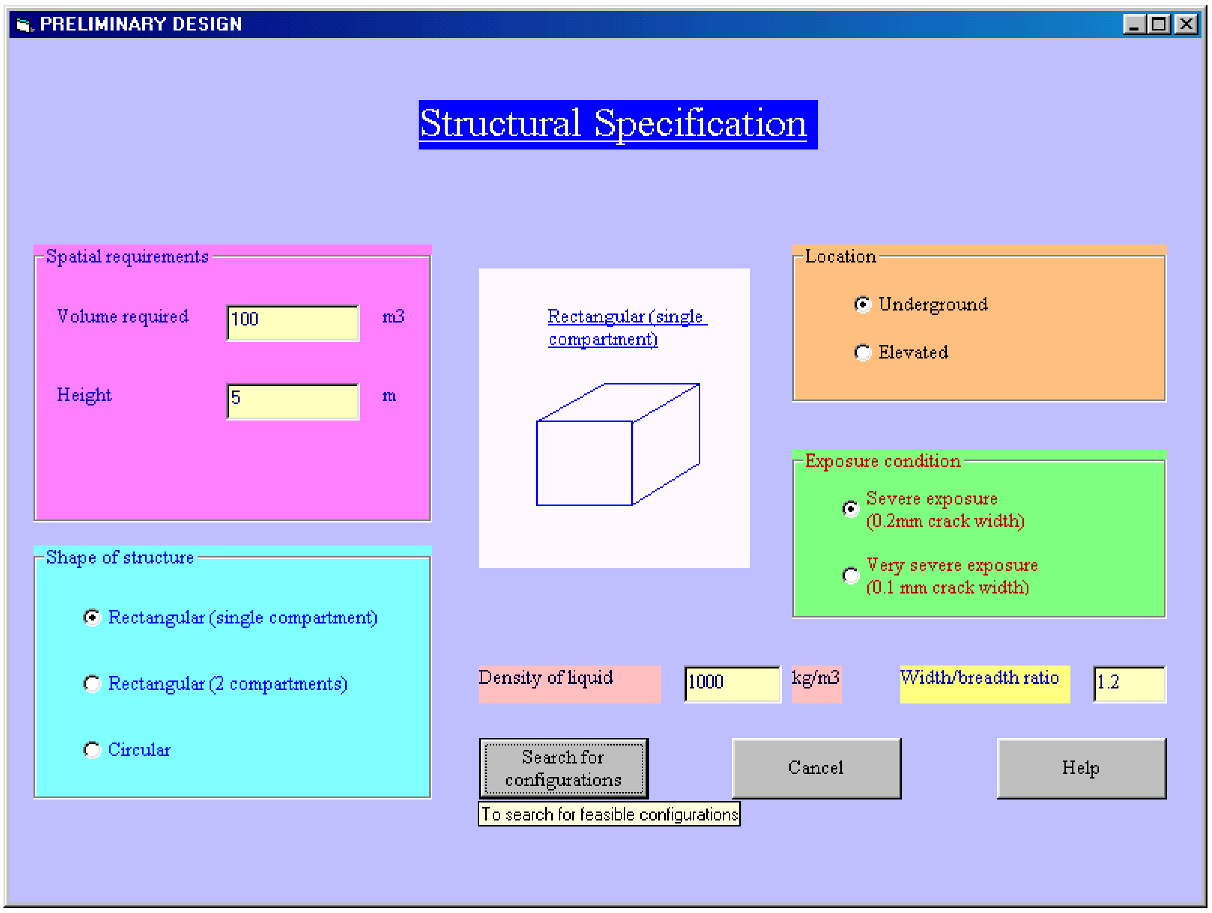


Figure 3. Structure of Visual Studio Components


PRELIMINARY DESIGN

Structural Specification

Spatial requirements

Volume required: 100 m3

Height: 5 m

Shape of structure

[✓ Rectangular (single compartment)]

Rectangular (2 compartments)

Circular

Location

[✓ Underground]

Elevated

Exposure condition

[✓ Severe exposure (0.2mm crack width)]

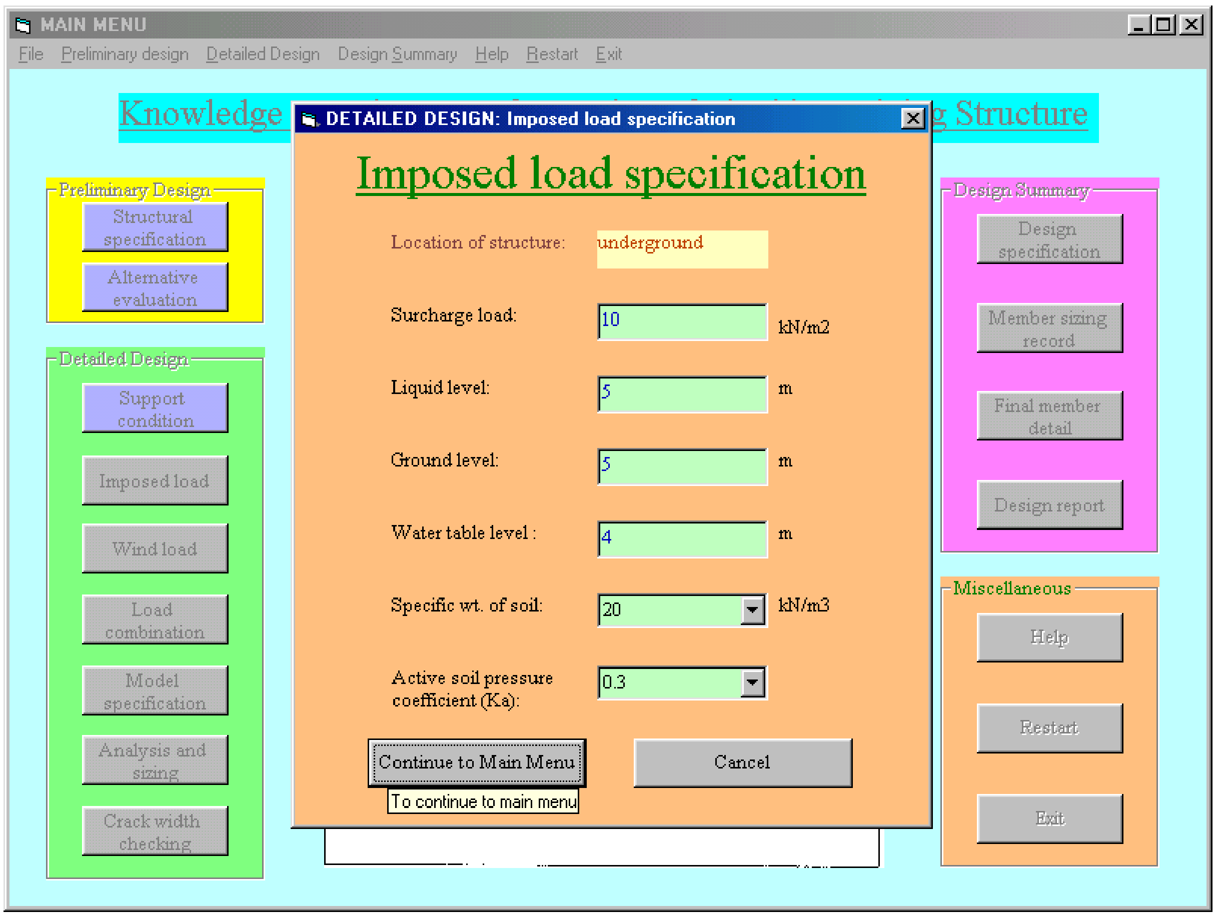
Very severe exposure (0.1 mm crack width)

Density of liquid: 1000 kg/m3

Width/breadth ratio: 1.2

Search for configurations
Cancel
Help
To search for feasible configurations

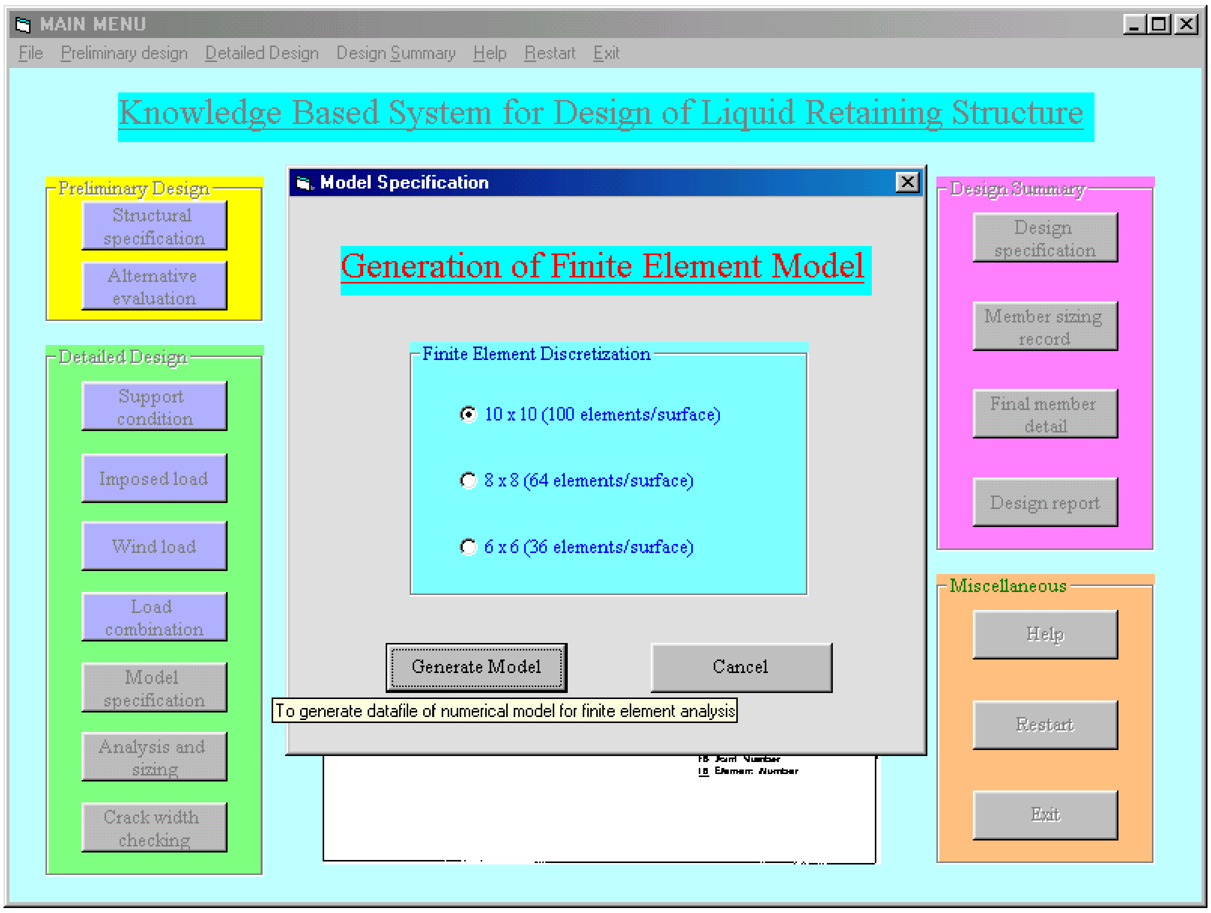
Figure 4. Screen showing structural specification in preliminary design



Imposed load specification

Location of structure:	underground	
Surcharge load:	10	kN/m2
Liquid level:	5	m
Ground level:	5	m
Water table level :	4	m
Specific wt. of soil:	20	kN/m3
Active soil pressure coefficient (Ka):	0.3	

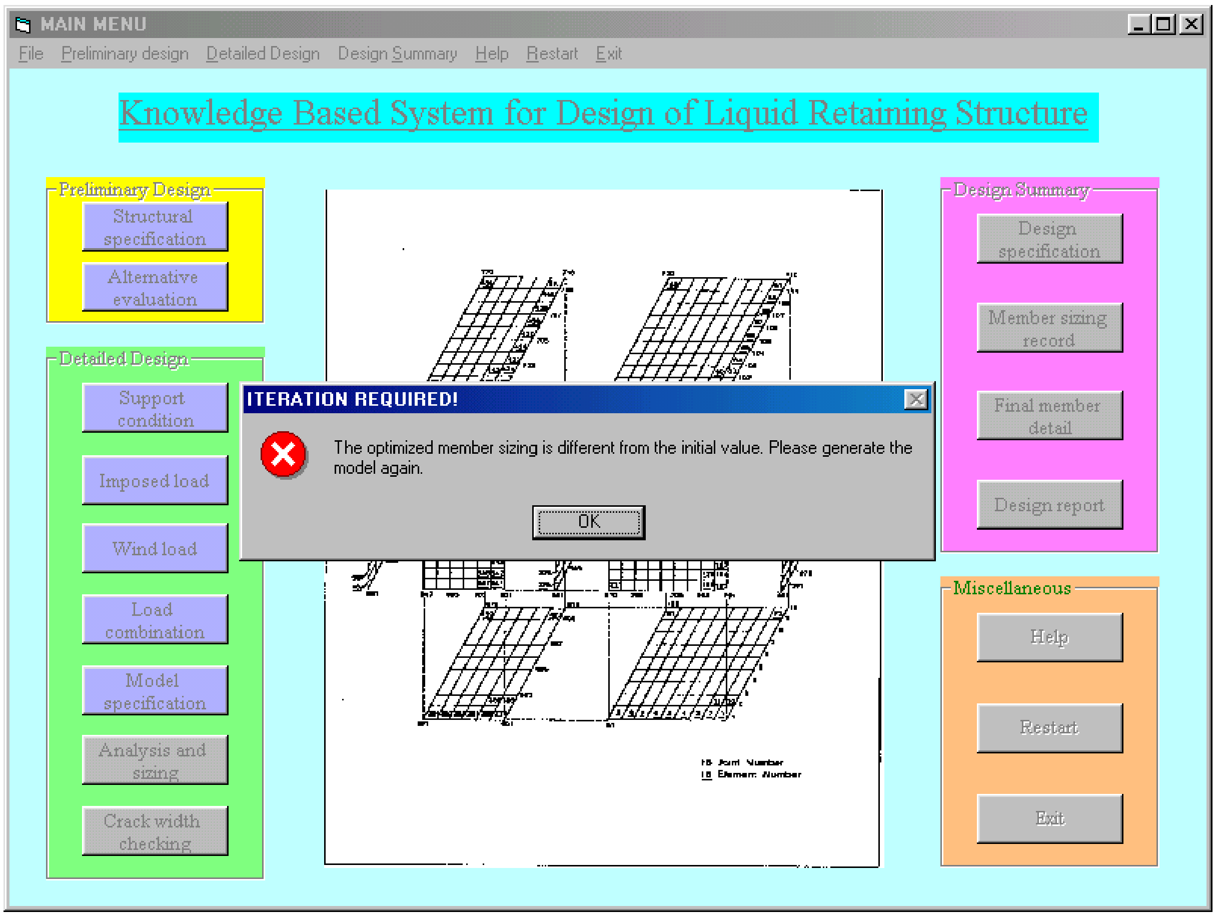
Figure 5. Screen displaying imposed load specification



Finite Element Discretization: [✓ 10 x 10 (100 elements/surface)] 8 x 8 (64 elements/surface) 6 x 6 (36 elements/surface)

To generate datafile of numerical model for finite element analysis

Figure 6. Screen showing generation of finite element model



The optimized member sizing is different from the initial value. Please generate the model again.

Figure 7. Screen prompting a message for iteration on model generation