INTEGRATION OF KNOWLEDGE-BASED EXPERT SYSTEM AND RELATIONAL DATABASE TO GRADE BUILDINGS FOR WIND RESISTANCE: DESIGN AND IMPLEMENTATION

by

SEEMANTINI P. GODBOLE, B.Engr.

A THESIS

IN

COMPUTER SCIENCE

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

Approved

Accepted

December, 1995

805
T3
1995
NO 152
cop. 2

JAC 2/12/96

ACKNOWLEDGEMENTS

I would like to thank all those who have helped me directly or indirectly in making this research successful. A special note of thanks to my husband Vinod for his support.

TABLE OF CONTENTS

ABSTRACT

This research was aimed at designing and implementing a system which would be capable of providing decision support, information management and easy to use user interface. Among many other challenges, one of the main challenge was to combine these different branches of computer science into one unified system. Some of the significant design features of this system include a loosely coupled design, run time binding of various components, relational database design and object-oriented user interface. It is also worth mentioning that this complex system with different components constantly communicating with one another, presents a unified front to the user. The user interface of this system was built with Microsoft Visual Basic, the database was implemented in Microsoft Access and the decision support component was built around the M4 expert system shell. The loosely coupled design facilitated a parallel design and implementation of each of the components. A user survey was conducted as a part of this thesis. The aim of the user survey was to gather information and feedback from the users about the system. This system is at work at the Insurance Industry for property Loss Reduction at Boston, MA.

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Insurance industry for property loss reduction insures various building structures which are classified as high rise buildings, heavy steel buildings, etc. While insuring, the insurance industry has to assess the building structure quality, its susceptibility to various natural and artificial disasters, and thus estimate the life of buildings and fix premiums. This assessment is very crucial as the millions of dollars are at stake.

The assessment, on one hand, is very crucial, while on the other, it is very difficult. The building could be in danger from various natural elements like floods, fire, wind (tornado, hurricane), earthquake, etc. There are a number of parameters to be considered as far as each of these elements are concerned. What the insurance industry needs is an opinion from an expert who can consider all the factors involved and give an opinion. The expanse of this problem is very wide and susceptibility of building structures to wind is a separate research area in itself.

At present, the insurance industry has some manual forms to be filled out by all of their customers. After getting back the completed forms, the insurance industry files the information after scrutinizing and sorting it. The retrieval of any of this information is done based upon the building identification number which is unique. In essence, the insurance industry is maintaining a database manually for managing the voluminous amount of information. The susceptibility of a building to wind hazard is decided by consulting various people in the insurance industry itself.

Regarding the above problem, the insurance industry approached the Wind Engineering Research Center (WERC) at Texas Tech University. The insurance industry needed a fresh approach to the problem. The general idea was to design a software tool which will make the process decision support and information management efficient and easy.

The WERC is carrying out a very active research in the field of "Wind Damage to Buildings." Research has been going on to identify the factors which affect the building's susceptibility to wind. The severity of each of the factors has been determined. Depending on these, very methodical and systematic solutions have been determined to grade the building structures depending on its susceptibility to wind.

However, it is practically impossible for these experts to attend each and every case presented by the insurance industry. This kind of situation seems to be needing an expert system. "*Basically, an expert system is a computer program that is able to model the decision making capabilities of human experts in a specific area of application. It allows novices to use experts' knowledge*" [17]. It is clear that a software tool with decision support capability is needed here. Considering the volume of information involved, the software tool cannot be complete without a database component. A careful design of user interface is required to satisfy the wide spectrum of users involved.

In essence, a complicated software system is under consideration here. Here the problem is of combining all these components into one single system. The integration of all the components should be transparent and the user should always feel that he/she is working on one single system. It is needless to say that getting components which would fit the requirements in all respects is impossible. The research is thus aimed at finding out about customizing certain components, designing the rest, and then integrating them into one unified system.

The following chapters in this thesis explain about other aspects of this research. The chapter on "Problem Statement" describes this problem in detail and outlines the requirements of such a tool. The chapter "Design Issues" states the various issues involved in design of such systems. It highlights all the problems and issues encountered while combining different components into one single system. The chapter on "Implementation Issues" concentrates on the implementation part of the research. "Literature Survey" states all the literature survey carried out in this research. Finally, the chapters "Field Survey" and "Results and Conclusions" discuss the outcome of this research.

CHAPTER 2
PROBLEM STATEMENT

This thesis is aimed at studying a problem where diverse areas or branches of computer science have to be combined into one efficient single system. This thesis suggests one of the approaches to combine components to build powerful but flexible application. This research is aimed at designing a system that combines the expert system technology, database management and provide a user interface that will give a feeling to the user of working on one integrated system.

2.1 Problem Areas

The following points explain the selection of an approach of integration and customization over all other approaches.

One of the approaches could be to use a general purpose expert system shell to load the knowledge base, which could carry experts' opinion, and answer the questions asked by the shell to get the conclusion or experts' advice. However doing so will introduce the following problems:

1. *The language of an expert system shell is often the language of the expert and is not well-suited for the casual, infrequent or novice user (e.g., an executive)* [11].
2. As the nature of expert system shells is very general, one cannot increase his/her domain knowledge and cannot learn anything about this particular problem of susceptibility of structures to wind.
3. Except for the building rating, the user does not get any advice or opinion as to how improve the rating of the building under consideration.
4. The options for storing data are very inefficient. Generally the user data can be stored in the text files, hence forming numerous text files, each file containing a set of data.
5. Some expert system shells provide very limited database support. If this database support is made use of, then the user has to work on a integrated system which supports both the expert system shell and the databases. This poses some additional

problems. For example, if any future enhancements are required in one of the components, the user has to switch to a new integrated system, which will be compatible with both the components, the old knowledge base and database.

6. Without databases, one cannot query the data. For example, one cannot find out the "average building grade for all the buildings in west Texas."
7. Lack of a strong database support makes it difficult to sort, search or manipulate data.
8. The user interface is the one offered by the shell and it cannot be customized. This is particularly not accepted by the novice users.
9. Using such general purpose shells is unacceptable to insurance industry as they need special kind of user interface, powerful database capability and most of all a system which will keep the intended group of users in mind. The intended group of users is executives, insurance agents, novice users, etc.

After reviewing the above approach, it is clear that the system to be built cannot have only the expert system or database component. The system to be built should perform information management through the database and decision making through the expert system. The system should somehow find a way for the two components to communicate with each other. In order to give a feeling of one integrated system it should have a user interface which will do so.

2.2 Requirements of the System

It is extremely important to understand the system requirements in any software project. In this case the following requirements were established.

1. The intended group of users for this system is very diverse. It is clear that the intended group of users are not knowledge engineers or experts but they are company executives, insurance agents, novice users, etc. The system should therefore be general in nature keeping in view the requirements of all the involved users. For example, this system cannot present a total database oriented front as it would be difficult for the expert system users.

2. If there is any change in the domain expertise or knowledge, then the system should be upgraded. In essence, if there are any changes in the experts' opinion they should be easily reflected in the system, without any major overhaul or changes in the system.
3. In general, the system should be flexible as need may arise to replace any component. The replacement should be possible without any major design changes. For example, the present expert system component may have to be replaced with some other more advanced expert system component.
4. The individual components should designed in such a way that they should be portable. Some users may want to give the database their own front end and carry out some statistical analyses on it. Some may want to use just the expert system component of the whole system.
5. The software system is expected to have an acceptable or average speed. The involvement of so many components and the communication between them should not slow it down excessively.
6. Though there are a number of components involved at run time, the users should not encounter any memory problems. This problem may be especially difficult to handle in DOS where the developer is constrained by the size of conventional memory.
7. Though the system would be made of components and would not be one integrated system the user should not feel it. The user should be unaware of the internal architecture.
8. The users are not expected to understand the language of expert system. The users are not expected to understand any error messages given by the expert system component and are not expected to be able to debug them.
9. The system should be easy to use and easy to navigate through and should have user interface according to the current industry standards. As the user group may not be very adept at using various software packages and hence special care should be taken while designing the human computer interaction part of this system.
10. The novice users should be able to increase their domain knowledge through this system. While answering a question an expert may know as to what is meant by "wind

speed region," but a novice user may want to get some more information. Some explanation followed by possible answers would help a lot.

11. While collecting the data from the user, explanation should be provided as to why this data is needed. This is in line with the requirements for expert system user interface. This would also give an idea as to why experts need this data and how much importance they pay to it. It gives user some insight into the decision making process and helps the user in understanding the system better.
12. The system should not only give the rating for the building under consideration but should also give suggestions that could improve the building rating. This would help the insurance agency as well as the customers to make improvements and lower the building's susceptibility to wind.
13. As the data collected from a user regarding individual building is quite voluminous, the user should be in a position to save all the data he has entered and should be able to retrieve it. This system should help the users in retrieving the data depending upon certain criteria. For example, system could provide some tools which could help users in retrieving all the buildings in Dallas, TX.
14. The system should be able to compare different building structures to help in users' comparative study. The users may want to know how all the buildings in New-York rate with respect to one another. Users may also want to do some statistical analysis on the building grades and may want to find out as to how many buildings fall within a particular grading range.
15. A user should be able to generate a comprehensive report on any building. The report should give all the details concerning the building characteristics and results. This will help in producing a hard copy on a particular building. This could help in sending building information over fax or mail. The users should be able to save the report as a soft copy which could be handed over to other users for further analysis.
16. It is required that the system should keep track of all the data and results regarding all the buildings. The data should be organized in such a fashion as to make it easy for the

user to sort it according to regions, cities, etc. This would help in retrieval, comparison, etc.

17. The user should be able to query the existing data. For example, the user should be able to generate a report of all the inferior roof structure buildings in Texas.

CHAPTER 3
DESIGN ISSUES

This chapter explains in detail all the issues considered while making the detailed design of this system. The various design issues regarding this system are given below.

3.1 Design Issues Regarding the Expert System Component

3.1.1 Knowledge bases versus Integrated If-Then-Else Rules

Decision support can be provided through several means, using an expert system shell is not the only approach available. The experts' opinion can be hardwired into the program. However, the following things should be considered before making this decision.

The idea of hardwiring the logic or experts' knowledge may not be possible as the expanse of this problem is very wide and the resulting logic, to say the least would be very complicated, prone to errors and hard to maintain. The other approach would be doing all these calculations through a separate DLL and then making calls to this DLL, as and how it is needed.

If the third approach to this problem is taken, the implementation would be quite different. The knowledge bases could be loaded through an expert system shell. Expert system shell could then be provided with the data regarding the building under consideration. The expert system shell would then have to carry out its consultation, where it takes the data regarding the building under consideration and analyzes it according to the expert's knowledge which is situated in the knowledge bases.

As the research in this field continues, the knowledge bases may have to be changed from time to time. Instead of rebuilding the executable every time, only the knowledge bases could be changed. These changes could be implemented by knowledge engineers or the experts themselves. They could just edit or change the knowledge bases. These changes would not require software professional's attention. Given the rate of change of technology, it may not be very wise to call upon a software professional every time.

In this way we would be introducing components in the expert system part of the whole system itself. However there are some problems in having knowledge bases separately. Once the experts' knowledge is in a separate file, it will open to anybody to access. Even if the files are stored in binary format, there are programs to decode them. The encryption of the knowledge bases have to be done in the framework of the expert system shell. In short, the knowledge bases can be encrypted only to such an extent that the shell can decrypt them. Hence, how much ever one might encrypt these, anybody else who is having the same expert system shell will be able to load the knowledge bases and decode them.

As already discussed in the introduction, it is difficult to find software components which would fulfill all the requirements. The above problem requires some basic changes and rebuilding of the shell which would make the shell unique. The solution to this problem is discussed in detail in Chapter 4 -- "Implementation Issues."

3.1.2 Run Time Binding versus Static Binding

If the expert system is going to be a separate component, then there could be static binding or run time binding through a DLL.

The advantages of building a DLL is that any changes to be made in the expert system shell require only the DLL to be rebuilt leaving the executable unchanged. The size of the executable will not be as huge as it would be in the case of static binding. Whereas in the static binding, the executable has to be rebuilt every time there is some change made in the expert system routines.

3.2 Design Issues about the Information Management

3.2.1. Information Storage -- Text Files versus Databases

As it is evident from the requirements, the user may want to store data for each and every building under consideration. All this data has to be managed, stored and retrieved whenever the user wants to do so.

Managing the data through text files is easy to implement. Each of the files can contain data regarding a particular building. The data can be read from or written to these files. However this way of data storage is not efficient. A bunch of small files can soon become unmanageable. This approach also yields to a very inefficient use of the disk space.

The biggest disadvantage of file approach lies in the failure to execute queries. It is just not difficult, but it is impossible to carry out queries on the data residing in these numerous small files. For example, to answer a query like "generate a listing of all the high-rise buildings in the state of Kansas that are graded lower than 4.5," the system has to open each and every file search for the particular line and generate a listing. This method is not only time consuming and inefficient, but there is every chance that the system might fail in the process of making so many disk accesses in the file operations.

Databases require more time in the design phase. In the case of this system, all the entities involved are implicitly related to one another. Hence in this case relational data model seems to be the perfect fit. In the design phase, relations between all the entities and attributes of all the entities have to be recognized. Some new products in the market have simplified the implementation phase of the database. Carrying out queries on the database is systematic, efficient and provided that the database is designed properly, queries yield accurate results. One more positive point of implementing the databases is that the design can be extended to allow a multi-user access. Locking of database can be implemented to maintain the database consistency. In this way multiple users can access the database. The same thing would be impossible to implement in the file approach.

3.2.2 Entity Relationship Model

The various entities and their attributes involved in this databases are as shown in the Figure 3.1. The data model is relational.

The basic entities in this system are namely "Building," "Surveyor," "Consultation," "Environment data," "Roof envelope data," "Wall envelope data," "Framework data," "Other miscellaneous data" and "Result." The "Building" is an entity which would represent the building under consideration. This building will have a "Surveyor" who will survey the building. The surveyor has his personal data, like name, number and building specific data like his comments on the building. A surveyor may examine more than one building.

A building has a "Consultation," every consultation has a unique set of building data associated with it. A building may have more than one consultation associated with it. For a particular building the user may just want to change some data items and observe its effect on grading. Every consultation has a set of "Environment data," "Roof envelope data," "Wall envelope data," "Framework data" and "Other miscellaneous data" associated with it. Every consultation also has a "Result" associated with it. The entity relationship diagram is as shown in Figure 3.1.

3.2.3 Table Design

From the entity relationship diagram, it is clear that that there will be a one-to-one relationship between the entities consultation and other entities Environment, Roof Envelope, Wall Envelope, etc. Instead of having different tables for each of these entities,

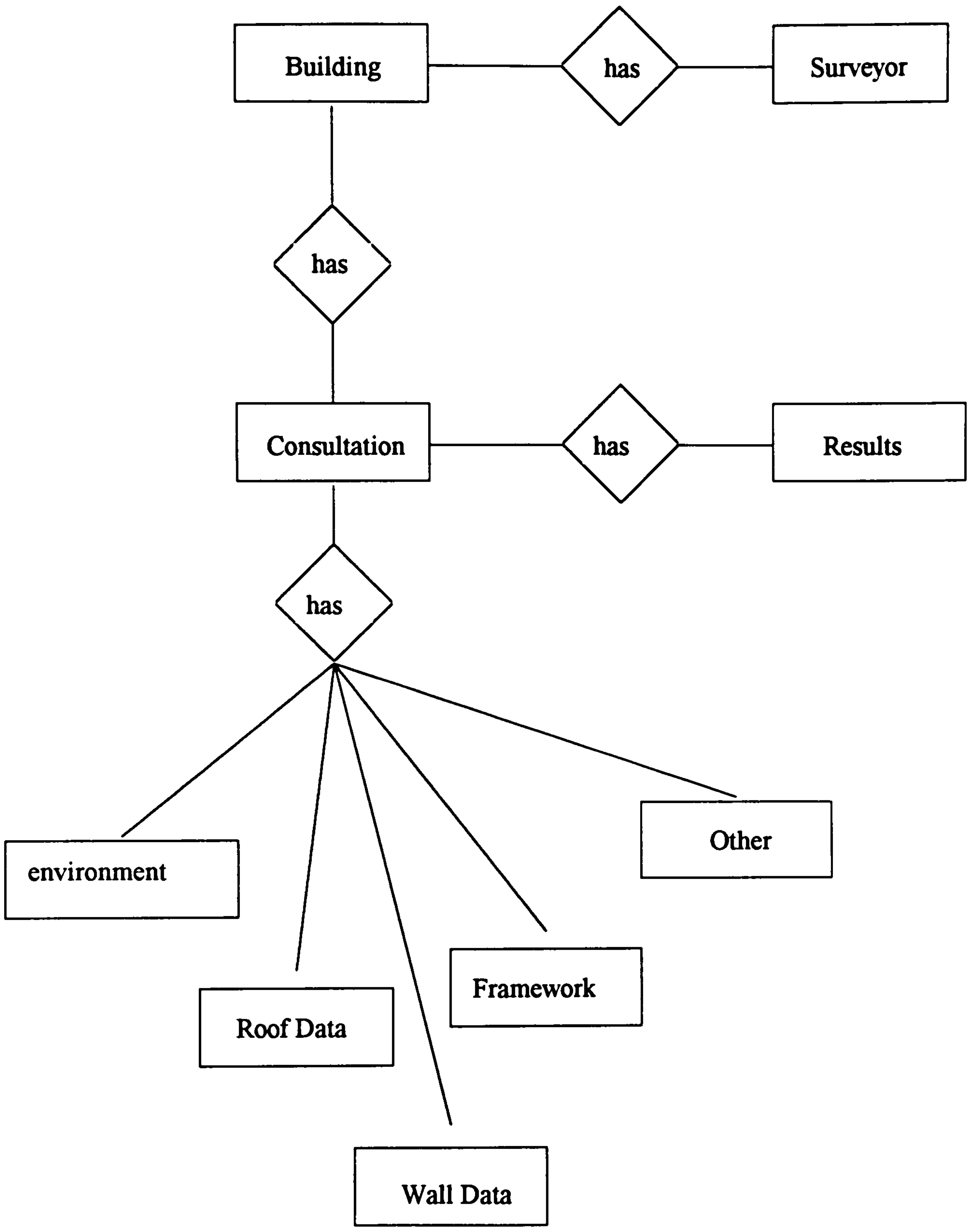


Figure 3.1 Entity Relationship Diagram

there could be only one table having composite key with components as consultation# and serial#. The latter identifying whether the record belongs to Environment or Roof Envelope etc. This would save the space for the rest of the table definitions.

However though the primary keys in all the data tables would be the same, the number of attributes are different. Only one table to store all these categories of data may save some table definition space but this will waste lot of space considering the unequal number of attributes for each of these data tables. Considering the volume of data involved in this system having separate tables would save more space and facilitate a better design.

3.2.4 Relational Schema

The relational schema of this E-R model would be as follows (Figure 3.2).

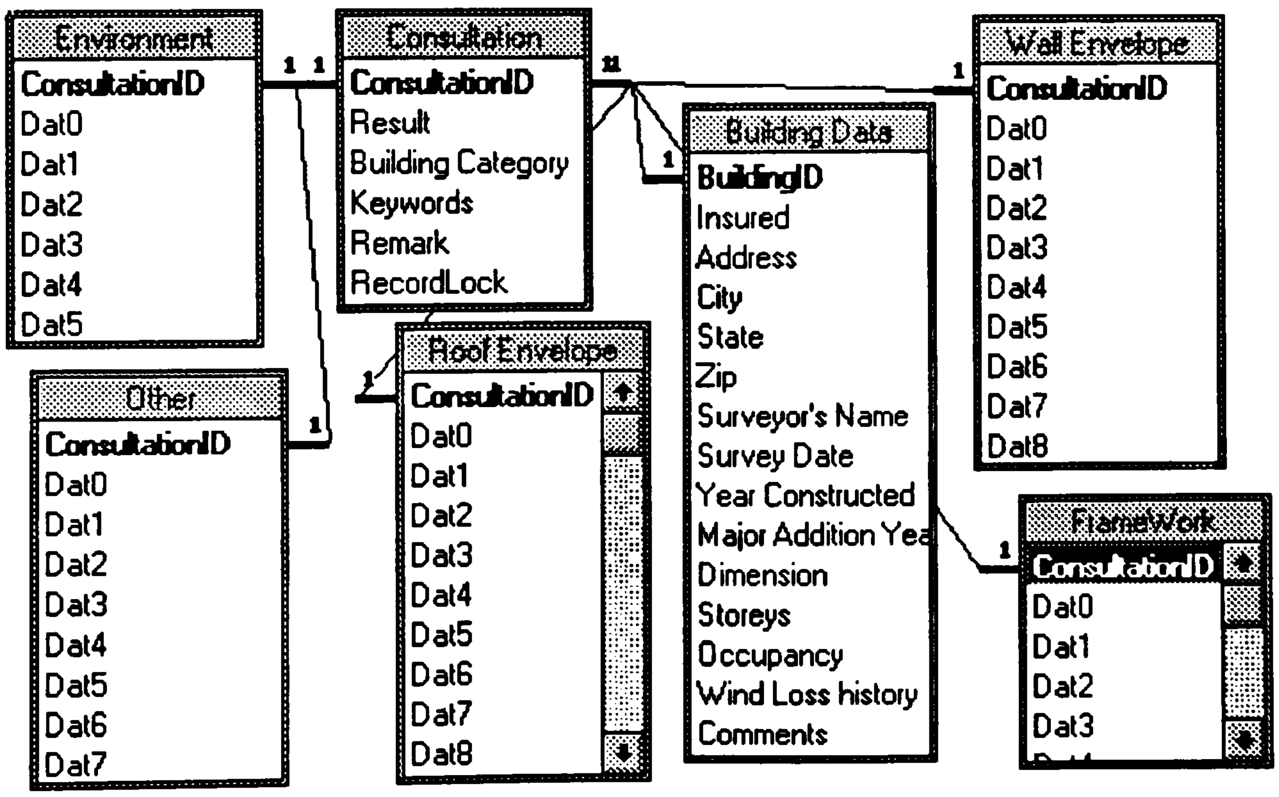


Figure 3.2 Relational Schema

3.2.5 Speed of the Application

As it is estimated that there would be number of database accesses while the user is working on this system, the speed of the application has become a separate design issue.

The steps taken to keep the speed of the application to an acceptable level on a reasonable hardware would be discussed in details in Chapter 6 -- "Implementation Issues."

3.2.6 Record Level Locking

In order to provide a multi-user access to the database, it would be essential to lock the record. Most of the database management systems provide some sort of locking to facilitate multi-user access. In the case of this system the data from the database will have to be copied into the system variables after closing it. The user may experiment with different things with this data and hence it is impossible to keep the record open for such along time. At the same time though the record has been closed it has to be locked as the user may manipulate the system variables and save it back to the database. The implementation of this aspect is covered in chapter 6 -- "Implementation Issues."

3.3 Design Issues Regarding the User Interface

3.3.1 Role of User Interface

In the recent years, there has been a growing awareness about the role of user interface in any software development. It has been generally suggested that the user interface design consists of 30-35% of operational software. In the system under consideration, it is certainly not going to be lower.

The aim of this research is not just giving a well-constructed database or excellent decision making performance but also to give a user interface that will let the user communicate with the system effectively. It is important that the user understands how to input or output the data or interpret the information. User should have a clear idea as to how the system will react. In other words the aim of this research is to make the user interface easy to use and understand so that the user can concentrate on the problem at hand rather than thinking about "how to get this system to work."

3.3.2 User Interface for Expert Systems

As the user interface is the central coordinator in this system, the success of this software system largely depends on its user interface. The user interface issue in case of the expert systems is quite complicated and needs a lot of thought and consideration. *"Unfortunately, interface design is more complicated than just putting up the windows on the screen"* [8].

The designer has to consider the intended group of users and their specific needs. *"The designer must concentrate on many aspects of usability, including a focus on users and their tasks, getting empirical evidence about effectiveness. The designer is forced to examine who is involved at each stage. What are their particular needs? How are these needs best addressed in the design of the system?"* [8].

While designing, due considerations has to be given to the fact that the users of expert system based applications are different from the usual computer users. *"The user community for expert systems is often different from those using editors, operating systems and other traditional systems"* [8]. The designer has to examine as to *"what are the special needs of these professional users who, despite being computer novices, are often experts in their own field of endeavor?"* [8]. Such users require a strong help facility for their system as far as operational and technical matters are concerned.

As it is evident from the requirements and the literature survey that has been done, the users of expert system based applications certainly have special needs. For an expert system user, just getting an answer is not enough, the user also deserves an answer as to why this conclusion has been reached or what he should do to improve the results. Users should also get an idea of the expert's opinion. A rough outline should be provided as to why experts are asking for this data and why and how important is it in the process? One should also be able to increase his/her domain knowledge through this system. This feature will be particularly important to the novice users. The overall goal of such systems is not just to give an answer but also a justification for that answer. The goal is to have a technical dialog or a question-answer session or a consultation between the system and the user.

3.3.3 Evaluation of the User Interface

It is also a design issue as to how the designer should evaluate this user interface. How does one demonstrate that the designed interface is acceptable and beneficial to the users? This question is best answered in Chapter 7 -- "Field Survey."

As more and more applications are being developed in Windows environments, clear standards regarding user interface have been established. The intended group of users is familiar with the Windows interface, as the most commonly used packages are developed for a Windows environment. Users work on word processors, presentation software, spreadsheets which were developed for Windows environment. Due to the uniform pattern in the user interface, the users do not have to spend time in learning to navigate through the system.

3.3.4 Design Issues Regarding the User Interface Standards

As the user interface provides a front end to both expert system component as well as the database component, it is hard to follow the standards for either of them. As every software system is unique, it may not be possible to stick to particular standards. In this case, the user interface has not been made according to any specific standards but general guidelines have been followed which are set for expert system and database applications.

3.4 General Design Issues

3.4.1 A Windows/Non-Windows Application

The system can be implemented in both Windows or non-Windows environments. However, as more and more applications are moving towards Windows platform, very specific user interface and other software standards are being set. Even users find a lot of Windows user interface features amenable.

Memory requirement also plays a very important role in this design issue. There are lot of different components involved in this system, viz. the expert system shell, the database component and the user interface. All these components take their own run time memory space. While running under DOS, the 640KB conventional memory may not be enough. The expert system shell itself takes 200-250KB runtime space. Hence the main issue is that of implementing the system in DOS with a DOS extender or implementing the system in Windows environment. Considering all the above creating a windows based system seems to be the right alternative.

3.4.2 A Loosely Coupled System versus Integrated System

The main components of this system are going to be:

1. The Expert system shell
2. The databases
3. The user interface through which the user communicates with the system

Two approaches could be adopted while combining the above components to make a unified system. "*On one hand there are systems where the components are 'loosely' integrated by communication links among them. On the other hand there are systems where the components are 'tightly' integrated with each component serving as a subsystem in the larger system*" [11].

In the tightly coupled system, the components cannot exist on their own and cannot be executed separately. There will be one integrated system, which has capabilities of decision making, data storing and providing a user friendly interface. This kind of system if built can serve the purpose and also presents the front of a unified system to the user. However the programmer has to take care of all the minute details while implementing the system. These kinds of systems are more difficult to maintain and debug as it is difficult to point out the problem area.

On the other hand, a loosely coupled system can be built by combining stand-alone components. The expert system shell with its limitations can still be executed as a separate application. The same is the case with databases. However one more component can be built to communicate with expert system shell, databases and the user himself. This component could very well be the user interface component. The system can be represented as shown in Figure 3.3.

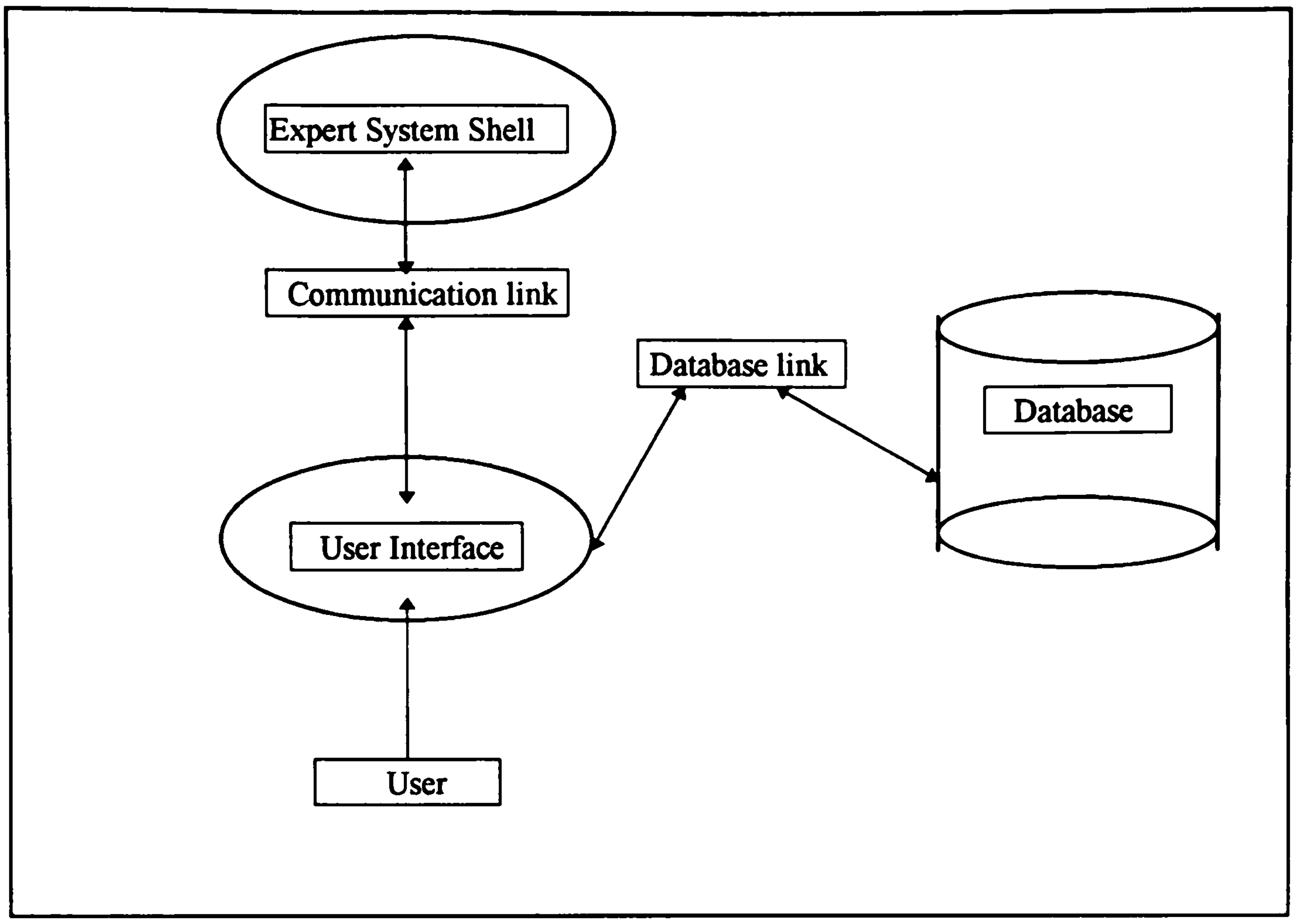


Figure 3.3 A Loosely Coupled System

The advantages of this system are very clear. As the components are distinct, maintenance and debugging can be easy as the defects could be pinpointed and corrected at the component level. The other most important point is that any of these components can be replaced or upgraded without affecting the other components. In the case of such a change, only the interface between the two components may have to be changed.
If the system is loosely coupled, the components could be ported to any other system for other research purposes. For example, as the database is going to contain very useful information, the insurance industry could port it to some other application to do analytical studies. Same would be the case with user interfaces. More detailed discussion in the subsequent sections would show that the user interface has been built with some templates. These templates could be useful to some other applications.

3.4.3 Design Issues Regarding the Architecture

After considering all the above issues, it is worth going into additional details regarding the interconnections of all the components. It is clear that the user interface has to be the central coordinator in the system. It has to collect all the data from the user or from the database and if requested, has to store all the data back to the database. Then it has to pass all the data to the expert system. It is an issue to consider whether the expert system shell should access the database directly. However, expert system shell may not have the capability to access the database or may have a very restricted capability to do so. To take full advantage of the capabilities of the database systems, it is best that the user interface communicates with the database and passes on the data to the expert system shell for further processing.

The user however does not know that there are so many components inside this system. He/she views the entire system only through the user interface and carries an impression that he/she is dealing with one unified system. This is very important keeping in view the intended group of users.

The system now finally would look as in Figure 3.4.

3.4.4 Usage of Object Oriented Concepts

The numerous advantages offered by the object-oriented technology, like encapsulation, inheritance, overloading and polymorphism, lead to a better software design. Inheritance can come handy while creating a template for user interface and then

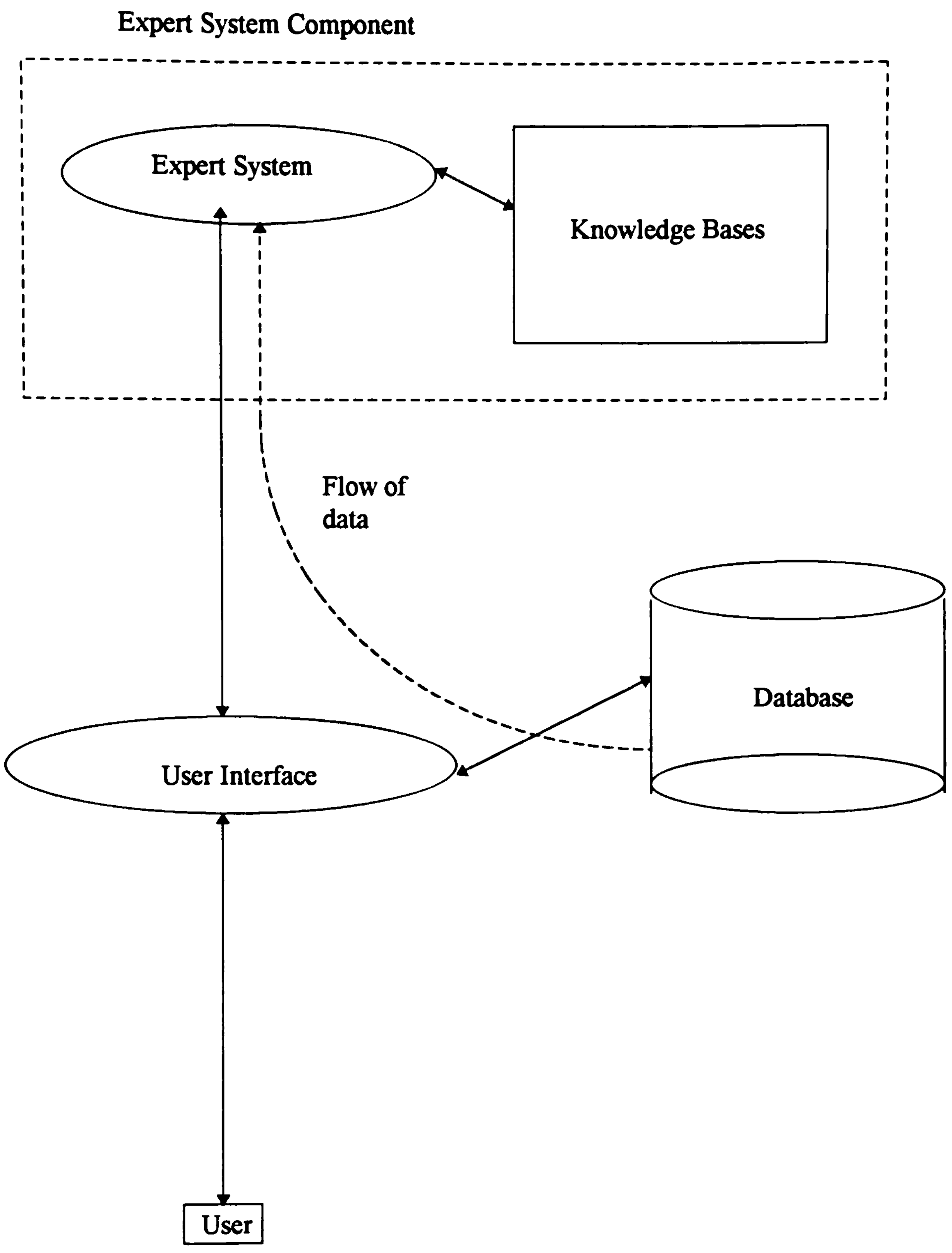


Figure 3.4 System Architecture

deriving all other classes or screens from this basic template. The usage of object oriented concepts for this thesis is explained in details in Chapter 6 -- "Implementation Issues."

3.4.5 Representation of this software cycle

Due to the fact that this system is built with loosely coupled components, the design and implementation of these components were done in parallel. The representation of this cycle is as shown in the Figure 3.5

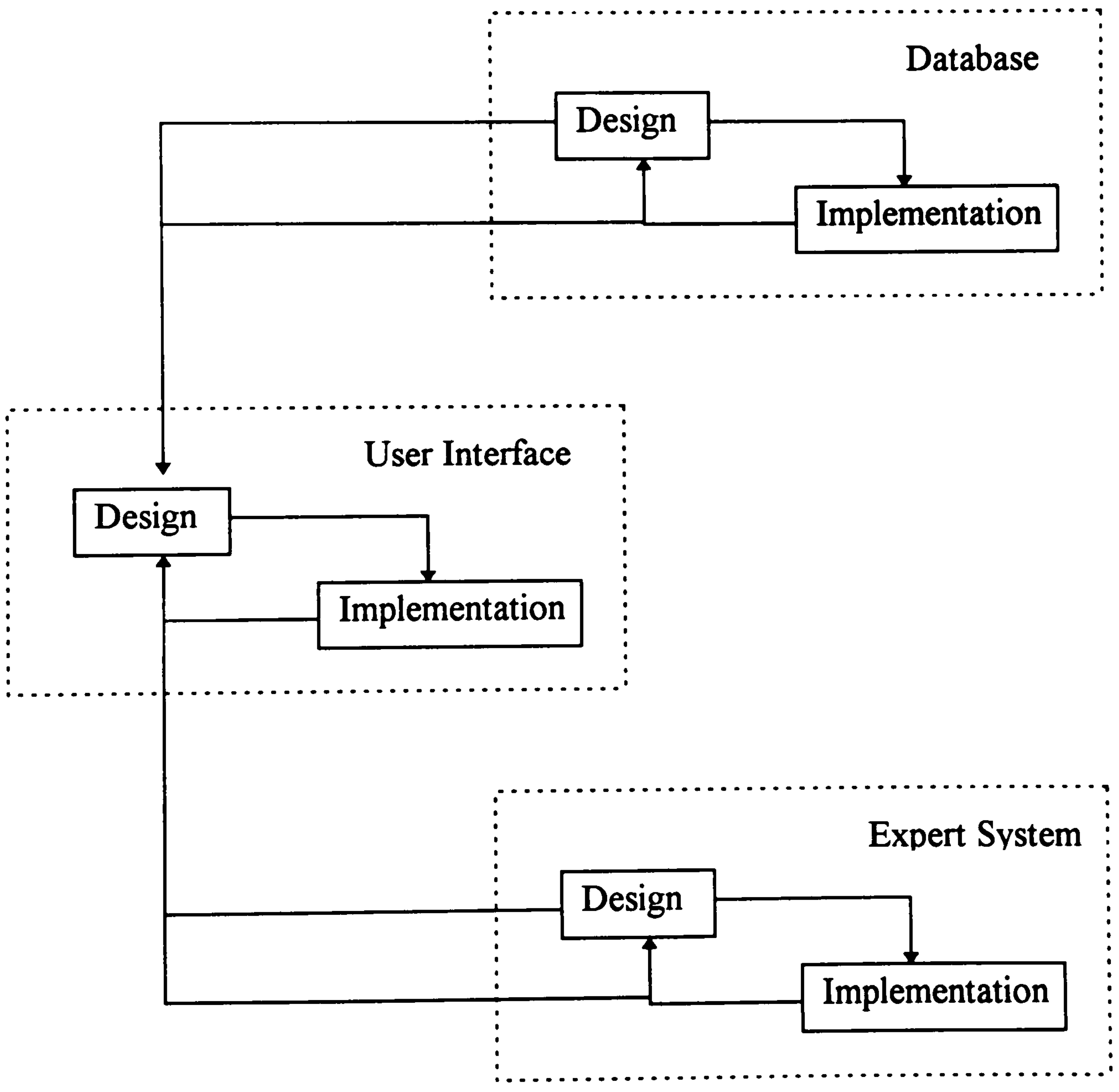


Figure 3.5 Part of the Software Cycle

CHAPTER 4
LITERATURE SURVEY

4.1 User Interface and Its Standards for Expert System Based Applications.

User interface for any application in general, plays a very important role. If the interface is ineffective, the system's functionality and usefulness are limited; users become confused, frustrated, and annoyed; developers lose credibility; and the organization is saddled with high support cost and low productivity.

According to New Directions in HCI [22], unlike the technology of industrial revolution, the technology of computing and communications is disseminated extremely quickly and at relatively low cost. As a result, one of the limiting factors on the use of new technologies is no longer the cost of production or distribution, but is the cost of learning and using the technology. Thus if the new technology is to be successful, mass access will have to be enabled by the development of useful, usable, simple, and extensible interactive technology.

James Hendler and Clayton Lewis [8] compared expert systems with more traditional computer systems and discussed the special needs of the former systems. Some of these issues have been discussed in Chapter 3 -- "Design Issues."

According to Dianne Berry and Anna Hart [1], in the recent years, there has been a growing awareness regarding standardization of the user interface in the expert system application development. Those in favor of standardization argued that standards in this area will result in more usable systems. While those against it argued that standardization is neither practical nor desirable. The authors reviewed both the sides and concluded that guidelines are more appropriate than standards for user interface design.

Marilyn Stelzner and Michael D. Williams [8] gave a pointwise interface requirements for expert system based applications. It stated that the interface should represent the domain user's natural idiom. The interface should also provide immediate feedback to the user on the effects of changes to the system state. The user must be able to recover easily from trying different alternatives.

Laura Euler, Eric Maffei and Adam Rauch [5] gave details regarding building of graphical user interface in Microsoft Visual Basic. According to them, Microsoft visual basic makes programming in Windows easier. It is a new development environment that allows you to create real Windows application quickly and easily. Visual basic abandons the standard, linear programming style of basic in favor of event-based model of windows programming.

4.2 Expert System Based Applications and Databases

According to Elmasri and Navathe [4] , the database is a collection of related data. The following are the main characteristics of the databases. A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot be referred to as a database. A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested. A database represents some aspect of the real-world. It can be of varying size and complexity. A database management system (DBMS) is a collection of programs that enables users to create and maintain the database.

Elmasri and Navathe [4] stated that there are plenty of advantages of the database approach over the traditional file processing approach. The advantages stated are as follows

In the traditional file approach, redundancy in defining and storing the data results in wasted storage space and in redundant efforts to maintain common data up-to-date. In the database approach, a single repository of data is maintained that is defined once and then accessed by various users.

Database systems are general nature, they are meant to work with any general database. Whereas the file processing software can access only one specific database. If there is any change in the database structure all the programs in the file processing approach have to be changed. Moreover the database approach is flexible, takes less development time, and can be kept consistent to give up-to-date information.

D. Ruiu, R. Divi, P. Katzberg and J. Katzberg [15] explained how the combination of expert system and relational database improve planning and productivity. They stated that the use of computer programs has resulted in the largest gain in work productivity in the utility planning domain over the last thirty years. Further improvements could be obtained by solving a series of problems associated with creation, access, storage, and maintenance of input data for these programs. The authors have stated that such a problem could be solved by introducing databases for management of the data with expert system to draw conclusions. Though they have particularly written about electrical utilities, the above is applicable to any industry.

Larry Kerschberg [10] mentioned EDS, the expert database system. He states that expertise may reside within the system to improve performance by providing intelligent question-answering, using database semantic integrity constraint for query optimization. It further states that present day enterprise considers data as a corporate level resource to be managed by DBMS (data base management system). Organizations are realizing that the knowledge-based applications can serve as a mechanism for competitive advantage by providing reasoned advantage for decision making. Thus, to solve the problems data and knowledge go hand-in-hand, and it behooves the organization to manage both.

H. Craig Howard and Daniel R. Rehak [7] talked about the production expert system. They state that such kind of expert system applications require to process voluminous data and will require to access large distributed databases. The need to use expert systems when addressing larger problems is the motivation for developing expert system-DBMS interface. Authors also shed some light on the differences between a business database and engineering database. It highlights the problem that in case of a engineering database it is difficult to anticipate all the eventualities or to anticipate all the possible queries.

There are several such articles and material which emphasizes on the importance of interfacing expert system and the database management system highlighting that the combination improves productivity, efficiency and performance of the entire system.

4.3 Relational Database

Elmasri and Navathe [4] suggested that the relational model represents the data in a database as a collection of relations. Informally, each relation resembles a table or, to some extent, a simple file. When a relation is thought of values, each row in the table represents a collection of related data values.

As given by [4], a relation schema R, denoted by R(A1, A2, ..., An), is a set of attributes R = {A1, A2, ...,An}. Each attribute Ai is the name of role played by some domain D in the relation schema R. D is called the domain of Ai and is denoted by dom(Ai). A relational schema is used to describe a relation; R is called the name of this relation. The degree of a relation is the number of attributes n of its relation schema.

An example of relation schema given by the authors, for a relation schema for a relation of degree 7, which describes the university student, is the following: STUDENT(Name, SSN, HomePhone, Address, OfficePhone, Age, GPA).

CHAPTER 5
OUTLINE OF IMPLEMENTATION

5.1 Study of Expert System Based Applications

The literature survey was done on various expert system based applications The design and implementation issues concerning expert system based applications were identified from this study. It was discovered that the main components of this system were going to be the expert system shell, database and the user interface. Expert system shell was identified as a decision making component. Database was required for storing and managing the data and the user interface was brought into the picture for customizing the application. Several other stubs would act as communication links and coordinate with all the components.

5.2 Selection of the Expert System Shell

Selection of an appropriate expert system shell was a major step in this research. Finding out a suitable expert system shell is a research topic in itself. Selection of the expert system shell was done with the help of domain engineers and the experts. The main task was to select an expert system shell which could be linked through a run time library or which could act as a DDE server. The expert system shell was also required to accept the data on-line and off the line. The customized user interface could collect the data organize it into file and then pass it on to the expert system shell.

This way the user will not have to be bothered about the correct syntax or format. It was also required that the user need not posses any knowledge about how expert systems work and hence was not required to understand the error messages. It was thus desired to find an expert system shell which would communicate off the line.

The Cimflex Technowledge, a software organization in Palo Alto, California, has designed a expert system shell known in the market as "M.4". It meets the above requirements. It provides a DDE server as well as a DLL to communicate with it. This expert system shell collects all the data from the user on-line but also has a facility to accept a data file in a particular format. The results of the consultation can be received by the user interface and in turn can be presented to the user in the required manner. By programming it in a particular mode, the expert system shell would never interact with the user directly, but all its communication with the user will be through the user interface. In essence, the architecture would look like the one in Figure 5.1.

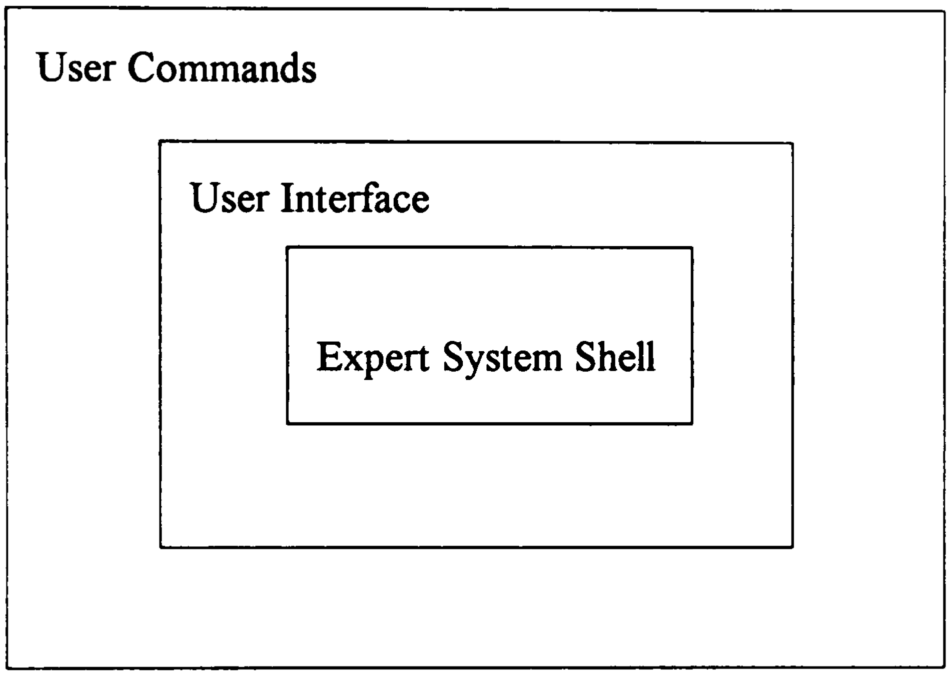


Figure 5.1 Users' View of the Expert System Shell

5.3 Selection of the User Interface Platform

As already discussed in the design issues, building a windows application is a better option as all DOS applications have conventional memory constraint of 640K. As pointed out in the New Directions by National Science Foundation [22], "*current interaction technology appears to have reached a plateau with the window-icon-menu-pointer paradigm that forms the basis for virtually all contemporary interfaces.* Keeping these points in view, it was decided to select an environment to develop windows application."

Recently there have been some windows application development environments which let you create user interfaces visually. Among them the main choices were Borland

4.0, Visual C++ and Visual Basic. From these environments Visual Basic was selected for the following reasons

1. The expert system shell library and DLLs are built in Microsoft and do not have good compatibility with programs written in Borland.
2. Microsoft Visual Basic and Microsoft Access have the same database engine. Thus it would be easy to develop databases in Microsoft Access and then link them with Visual Basic.
3. Though both Visual C++ and Borland C++ inherently yield to object-oriented design and implementation, the functionality of Visual Basic is also largely dependent on objects. The two main classes of Visual Basic objects are forms and controls. Every object has properties which actually in an object-oriented paradigm are attributes of that object. All these properties are referred to as object.property, a familiar concept for object-oriented programmers. The properties of all the controls are generally public, whereas procedures are private.
4. Visual Basic is a powerful platform to create a Windows application with relative ease. This platform inherently gives lot of flexibility and importance to user interface which seemed to be a vital part of this research.
5. Visual Basic adopts an event based model of windows programming which helps in making a better design.
6. Visual Basic can be extended by making direct calls to DLLs. It supports the DLLs that use the far Pascal calling sequence. Thus it can call the Windows API directly.

5.4 Details Regarding Linking of Shell and the User Interface

As already mentioned the expert system shell has a DDE server as well as a DLL. As the user interface has to pass the entire input data file and also the knowledge base to the expert system shell, run time binding with DLL seemed to be the better option. The DLL can be included in the Visual Basic project and then the user interface can communicate with it. It is not necessary that the user interface and the expert system shell be in constant

communication. Only when the user wants to carry out a consultation, i.e., evaluate the building, the expert system shell has to be initialized and communicated with.
The procedures in the DLL can be directly called by giving "API_EXEC" calls from within Visual Basic. All these functions have to be declared in order to let Visual Basic do the parameter checking.

The process of carrying out a consultation is roughly as described below. First of all, the system makes sure that the user has answered all the required questions. If not, it gives this information to the user. The user is then expected to fill out the rest of the question and try to start consultation again. Once this is ensured, the system initializes the expert system shell.

In order to communicate with the expert system shell, it opens input and output channels with the shell. These channels are the procedures which have to be written in the application which are called back by the DLL. Once the computation is started, the DLL starts processing and analyzing the data. Whenever a conclusion is derived or some errors are generated, the DLL calls the output channel and passes the results or errors as parameters. The functions in the system extract the result if the consultation was successful. Otherwise if there are errors, the system translates them in easy to understand language and presents them to the user. The flow chart for the process of consultation would roughly look as in Figure 5.2.

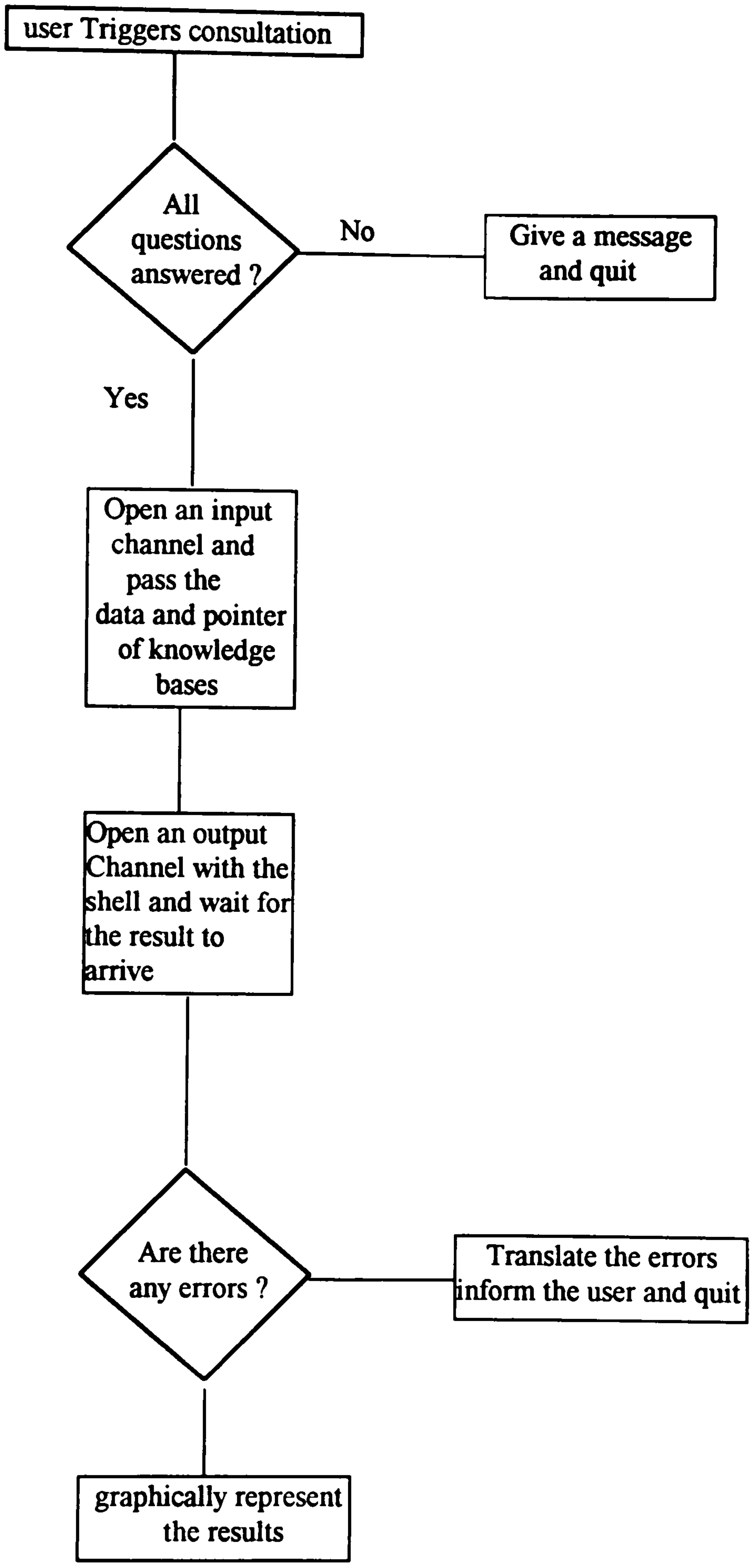

user Triggers consultation
All questions answered ?
No
Give a message and quit
Yes
Open an input channel and pass the data and pointer of knowledge bases
Open an output Channel with the shell and wait for the result to arrive
Are there any errors ?
Translate the errors inform the user and quit
graphically represent the results

Figure 5.2 Flowchart of Expert System Linking

5.5 Protection of the Knowledge Bases

The problem of protecting the knowledge base has already been discussed in Chapter 3 -- "Design Issues." The protection of knowledge base is essential not only from the point of view of security but also from point of view of their functionality. As it has been already explained, any user having the same expert system shell, or same expert system shell DLL to be specific, would be able to open these knowledge base. Once a user opens these knowledge bases, he has an unlimited access to the experts' knowledge. He also can edit the knowledge base or can add or subtract some facts from the knowledge base. As a result of this, the data expected by the expert system shell will become incompatible with data provided by the user interface. Thus the users will never be able to carry out a successful consultation.

A solution to the above problem would be changing the expert system shell DLL so that other users having the same shell but not the same DLL will not be able to open the knowledge bases. The expert system shell has the provision of saving the knowledge base in a fast-load format which cannot be edited or word-processed. However, anybody else having the same expert system shell can always load the knowledge base and view them. In order to solve this problem, source code of expert system shell had to be modified to provide a password protection. This modification makes the expert system shell DLL unique. Every time a knowledge base is saved through the shell, the new DLL attaches a user selected password to it. Every time the shell attempts to read these knowledge bases, the DLL looks for the particular password and decodes the knowledge base only if the password is correct. The unmodified shells with other users will not have the DLL which can decode this password and hence those shells would not be able to read the knowledge base.

5.6 On-line Help System

The system under consideration combines diverse fields like expert systems and databases. There may be many users who may understand some aspects of the system but may not grasp them all. Hence in this case the on-line help has to be very informative and easy to use. Help should cover all the information regarding the subject matter and operational aspect of the system. A tightly coupled help system has been designed to provide a search option as well as a context sensitive option. As the help is context sensitive, the user can focus on any control and press the F1 key to get help on it. This would serve to explain to the user the operational part of the system. A search option also has been provided where the user can browse various help topics, search for a particular topic and look at the related topics.

The help has to be rich in resources and in order to make users clear about the concepts, bitmaps, hotspot links and hypertext links are used. Thus, the user can jump from a topic to another topic. Novice users can increase their domain knowledge by searching for a particular topic.

CHAPTER 6
IMPLEMENTATION ISSUES

The following steps were involved in the implementation of this research.

6.1 Implementation Issues in the User Interface

Various special needs of expert system and database based application were being considered while designing and implementing the user interface.

6.1.1 Navigating through the System

In order to make navigation through the system easy even for novice users, the following steps have been taken. They also help the users understanding the system much better.

1. Selective enabling/disabling of menu items give an exact idea to users as to what they can or cannot do. For example, the record menu does not get activated until a database is opened or database menu does not become activated until a building category is chosen.
2. A toolbar with pictorial icons explains to the user the steps to be taken.
3. Tightly coupled help with context sensitive option and search options always assists the user in finding out the operational and subject matter aspects of the system.
4. Users would find some standard Windows features like file open/save dialog boxes or print dialog boxes very familiar. Similarly other standard Windows controls like list boxes, command buttons and text boxes would be self-explanatory.
5. Usage of MDI child forms makes it easy for the user to switch between different screens and find a way to answer a particular question.
6. The appropriate usage of modal and nonmodal forms indicates to the user the area on the screen to which he would be confined to take his actions or generate events.
7. Message boxes and warnings informing the users of their wrong actions help them understanding the system better.

8. The language of the system is easy to understand. However, some technical terms are particular to subject matter. Ample on-line help is provided to explain the technical terms which will help the novice user.
9. Changing the shape of the mouse pointer from normal arrow to hourglass indicates to the user that the system is at work and hence is not quick enough to respond to user requests at such a time.
10. Graphical representation of results helps the user comprehend the same very quickly. A user can also create a report which is self-explanatory in nature.

6.1.2 Getting Answers from the User

This was one of the important issues as this concerns gathering all the data from the user. This is crucial because this data is going to be passed to the expert system shell for evaluation. Any errors in this data will result in failure in evaluation. It is difficult because every user has his own distinct way of answering even while giving same answer to the same question. It is also essential that the system makes sure that user is providing a within the range and relevant information for the question under consideration. Looking at all this it was necessary to come up with the accurate and appropriate design and implementation. The two screens vital in getting answers from users are shown in Figure 6.2 and Figure 6.4.

The user gets a brief idea of the kind of questions he needs to answer from the screen in Figure 6.2. The text in the field named prompt just represents the keywords in the real question. This text gives sufficient idea to the user of the kind of data required. The user can click either on this text or in the answer space to pop-up another screen, which is shown in Figure 6.4. With this screen, the user can have a look at the real question and choices. From these choices, the user can select one and only one of them. He could then confirm or cancel his selection by clicking on the appropriate button. This takes the user back to the screen in Figure 6.2. If he has confirmed his selection, the choice appears in the answer space of Figure 6.2.

With this design, it has been assured that instead of typing an answer to a question, the user makes a choice which is most agreeable to him. Hence the language remains the same from user to user which avoids any syntactical or spelling error when this data is passed to the expert system. It has also been validated through experts and other users that the choices provided for each of the questions fairly encompass all the possible and relevant opinions. This arrangement also makes it impossible for the user to give false or irrelevant information.

In addition to all the above, two more features are provided. Consultation is not started unless and until it is made sure that all the questions have been answered. If an incomplete set of answers are passed to the shell, it would send an error message and try to get answers from the user on-line. This feature avoids such a situation. It has made sure that even if the expert system shell does not get a complete set of data it does not go back to the user asking for answers, whereas it generates an error message which can be handled by functions in the user interface and can be explained to the user.

6.1.3 Explanation for the Questions Asked

It is important for the user to know the reasons behind a particular question being asked. This makes the user aware of the experts' opinion and also it helps in understanding the importance of the question. Sometimes it may make the user more clear about the question itself or the context in which the question is being asked. Whenever the user views all the choices, he can also make a screen pop up, which shows him this reason. The same screen is used every time to pop up, but it shows the reason for the question under consideration.

6.1.4 Usage of Object-Oriented Concepts in the User Interface

There were a lot of screens having similar appearances and common logic behind them. To give a particular example, the questions falling in the same category were grouped together in a screen. All these screen had same kinds of controls in different numbers. The logic behind all these screens was the same. Similarly, all the screens showing the available choice as an answer to a question had the same controls but again in different numbers. The logic behind all these screens was also the same.

This seemed to be the right opportunity to apply the object-oriented principle of code reuse. A template could be designed and, depending on the need, a new screen could be instantiated from the template. The template could have just a sample of each control and more of those could be instantiated depending on the new screen. The whole code and sample controls could reside in the template and could be used by every new instance. In this way, the code could be reused simplifying a lot of things. As it can be seen from the diagrams, the screen which could look as the one in Figure 6.1 at design time would appear as the one in Figure 6.2 at run time.

Similarly the screen which presents choices to the user would look like the one in Figure 6.3 at design time and look like the one in Figure 6.4 at run time.

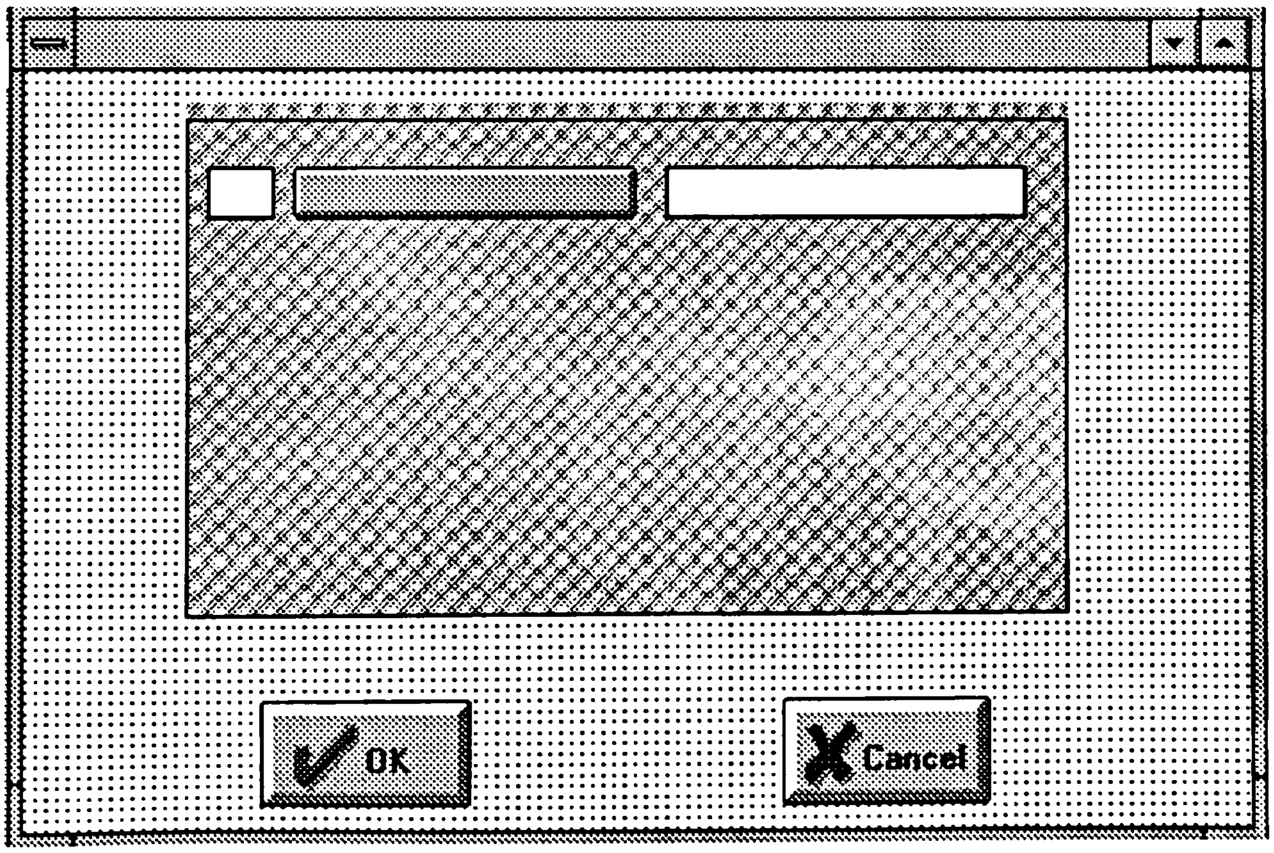


Figure 6.1 Template Screen

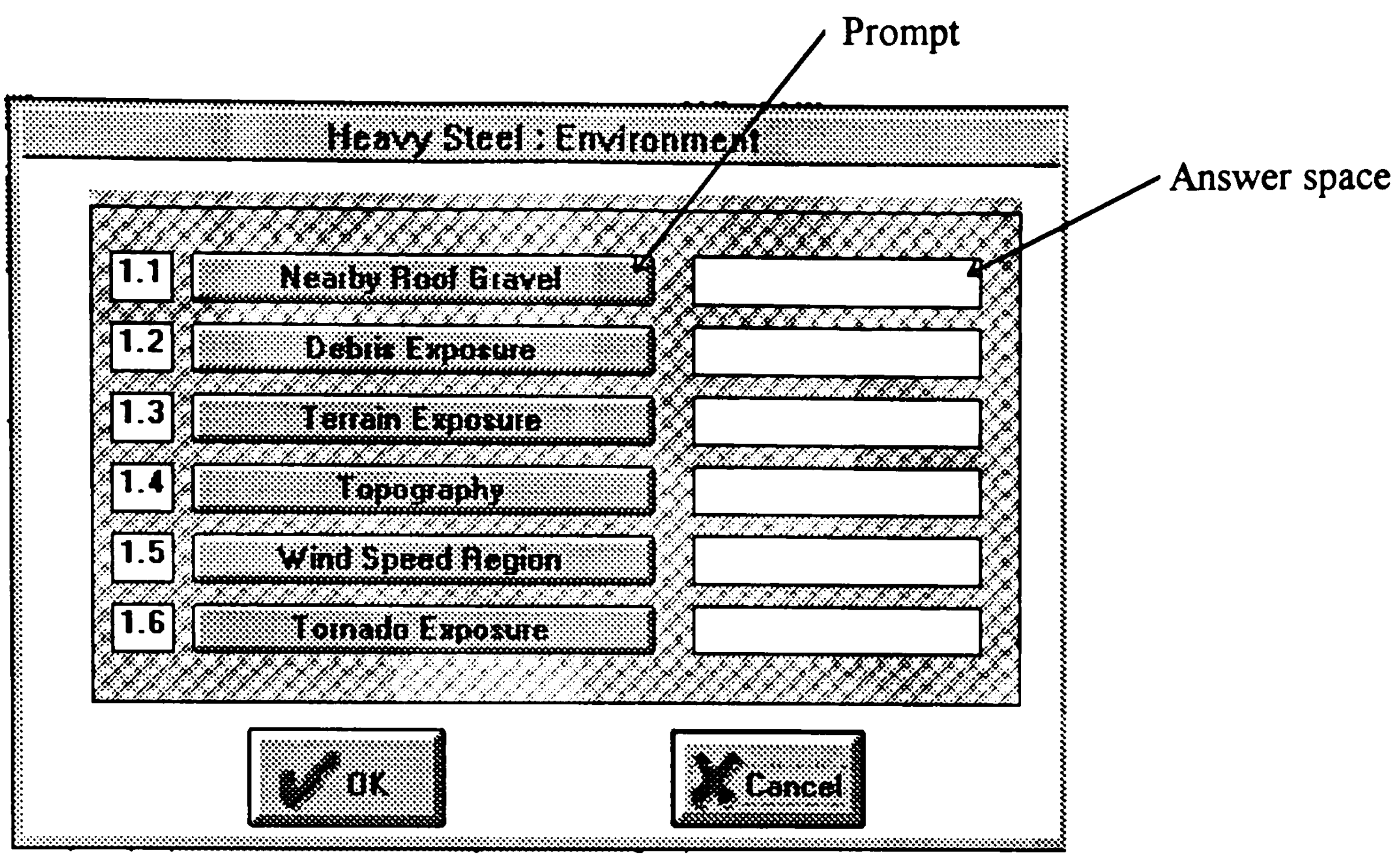


Figure 6.2 Screen at Run Time

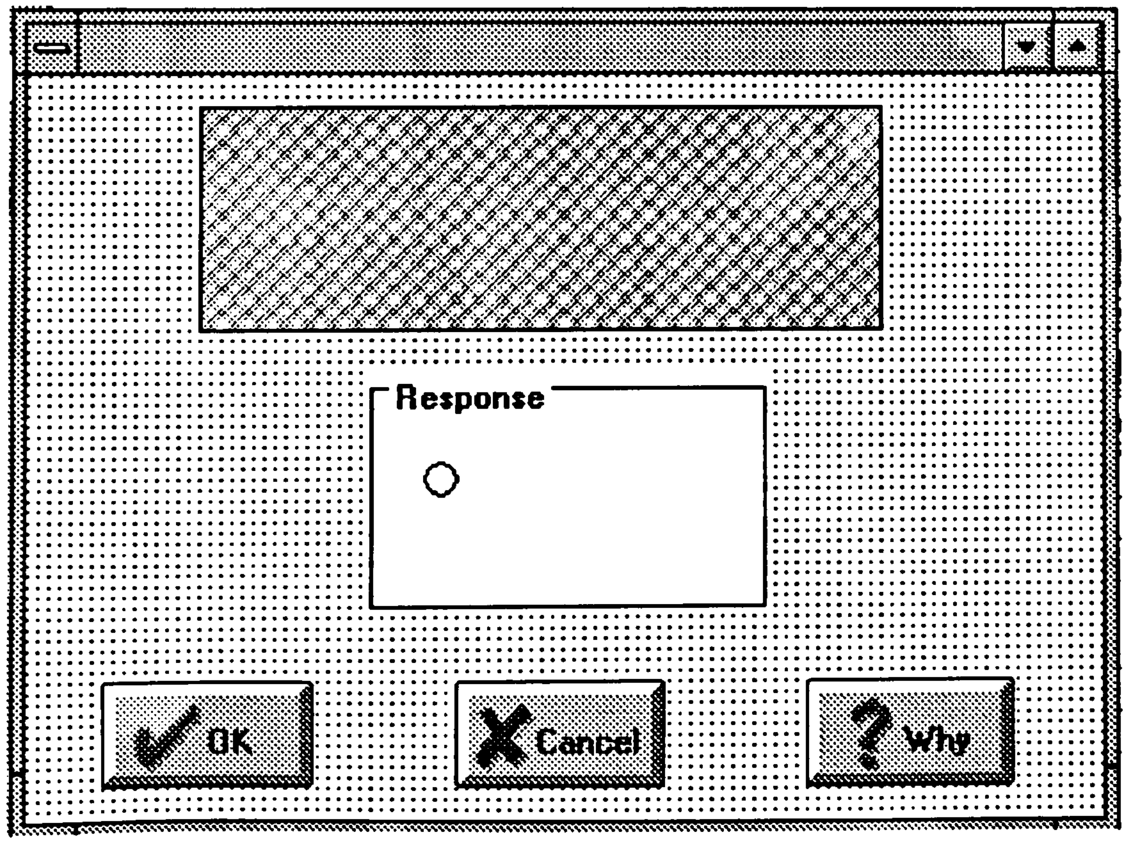
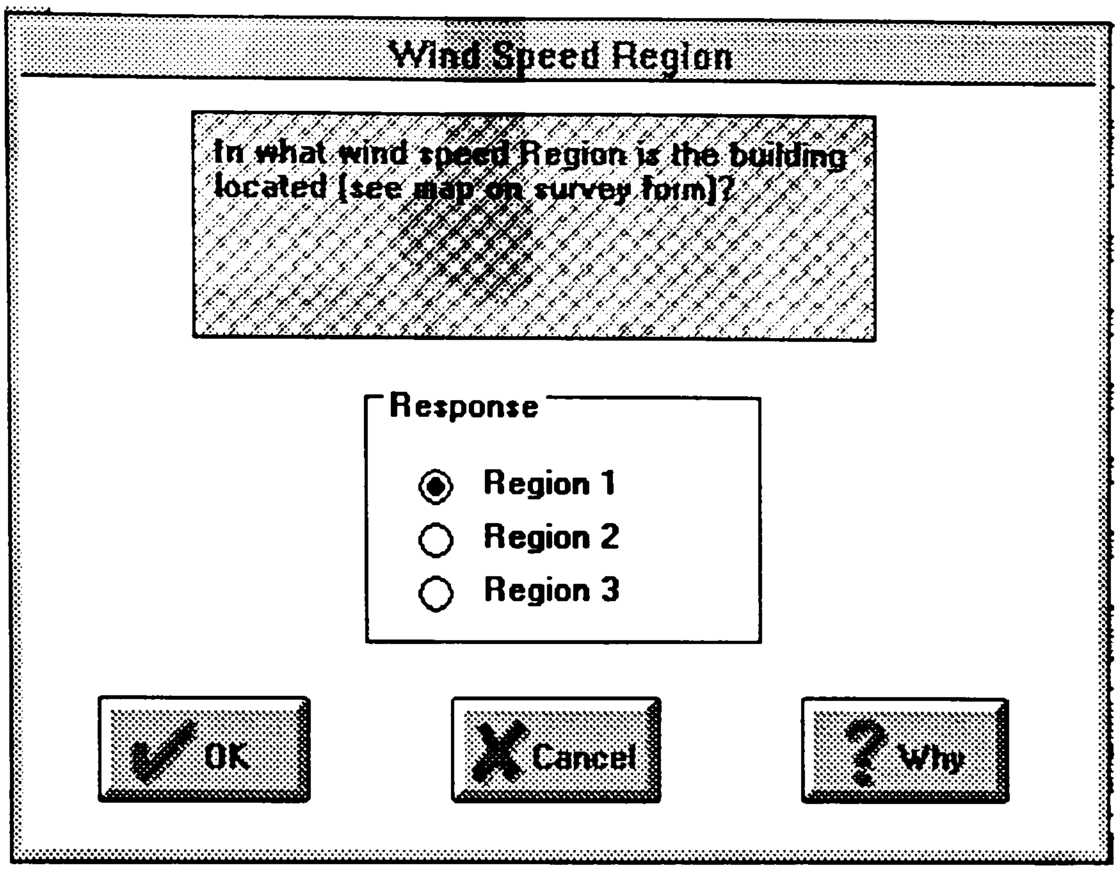


Figure 6.3 Another Template Screen


Wind Speed Region

In what wind speed Region is the building located [see map on survey form]?

Response

[✓ Region 1]
Region 2
Region 3

Figure 6.4 Another Screen at Run Time

The usage of these two templates and their instantiation at run time has reduced the number of forms from seventy-six to two. The advantages of reuse are as follows.

1. The code has been reduced as it is situated in only one template and reused by all the instances of the main template.
2. Any changes required in the future will have to be done only in one place instead of doing it in several places. For example, if all the forms are to be right aligned instead of left, it requires only the template code to be changed, all other instances will follow this code.
3. Very few resources are used at design time reducing the chances of running out of memory while coding, debugging and running the program.
4. The size of the EXE is reduced considerably as very few resources are used at the design time.
5. The process of compiling the code and making an EXE is quicker as the resources, code and the number of forms are reduced.
6. The whole code becomes easier to understand, debug and maintain.

7. This concept of reuse could be used in any general user interface where a base template could be designed and other screens could be instantiated from it.

6.2 Implementation Issues Regarding the Database

6.2.1 Implementation of Relational Database

The platform for implementation of database was Microsoft Access 2.0. Microsoft Access is a very powerful relational database management system. Users can create tables visually in Microsoft Access, they can write powerful queries, generate comprehensive reports and provide a front end user interface to it. There are other features in Microsoft Access which can compress databases provide individual passwords for them, etc. Working with such DBMS also gives some idea of the kind of user interface expected for database based software systems.

As Visual Basic and Microsoft Access run the same database engine, it was convenient to develop databases in Access and provide front-end through Visual Basic. For example, creating tables is more convenient in Access and manipulating that data is more convenient in Visual Basic. Development was thus done taking advantage of both environments.

6.2.2 Opening an Existing Database or Creating a New Database

A Windows standard common dialog box is used for letting users navigate through the file system and select a database to be opened. The database is opened with non-exclusive access letting multiple users open it at the same time. The locking is provided at the record level, which is described later, and hence the same database can be opened by multiple users. Once a particular database is opened, it becomes the current database giving access to all the records in that particular database.

The user has to specify path and name for creating a new database. There is always a empty template database which is copied on to this new path and filename. To begin with, this new database does not contain any records and hence there is no current record. The user has to begin by adding a new record.

6.2.3 Record Level Locking

In order to provide a multi-user access to the database, it is important that the current record is locked. Several DBMS provide this as a standard feature so that whenever the user tries to edit the record, the current page is automatically locked. Microsoft Access provides two types of locking schemes. One is known as pessimistic locking where the entire page containing the particular record is locked the moment user starts editing the record. The other is known as optimistic locking where the page is not locked unless and until an update on the record is performed. In both locking schemes, however, the entire page is locked which locks all the records on that page. Both schemes are not suitable for this particular system for the following reasons.

1. Users in this system may work on the data for a long time. The data may undergo a lot of manipulations and analysis and the user still may not save the data. It is not practical to keep the entire page locked with the pessimistic locking scheme for such a long time.
2. It is also not possible to adopt the optimistic locking scheme as the user is never sure of successfully updating the data in this scheme. This can be quite confusing to the users, especially if the system has to accommodate such a wide spectrum of users.
3. Due to all the above, the record is retrieved into system variables and the user is always looking at the data residing in the variables and not at the data in the database. On the other hand, it is also essential to lock the record so that the user is always sure that he is looking at the latest data.
4. One more reason for not going with the standard locking system is also because of the fact that at any point of time the user is looking at data which is a union of records, one from each of the tables. The standard locking practices thus will be locking a page from each of the tables, resulting in locks over a different number of records in each of the tables. Subsequent user wanting to open a different data set may be able to do so from some tables and may not be successful with other tables creating more confusion.

The relational schema for one of the most important tables, "consultation," is Consultation(<u>consultation#</u>, Building Category, Result).

As it is already known that the system determines the existence of certain record from this table, it will be useful to manipulate this schema resulting in one as below Consultation(<u>consultation#</u>, Building Category, Result, RecordLock). The new field "RecordLock," can be set or reset to mark the record as locked or unlocked, respectively. Even if the user has copied the entire data into system variables this record will always remain locked. The other advantage is, this scheme will get implemented at the record level and not at the page level. As soon as the user finishes his job with a record and tries to open a new record or add a record, the previous record is unlocked.

The locking scheme then could be implemented as shown in Figure 6.5.

6.2.4 Retrieval of Records

Considering the volume of data this system is expected to handle, it is important to design efficient search techniques. The user can hardly be relied upon to remember the complex primary keys. Hence the user should be able to retrieve the records based upon factors which can be easily remembered. For example, it is difficult for the user to remember that the primary key in record was "100KS12/3/94" but it is much easier for him to remember that this record was for a building in Kansas. The factors which are important for the users were determined and were made search criteria for retrieval of records.

Instead of designing different screens for each of the search criteria, only one screen was designed which would help in narrowing down number of retrieved records. The screen is as shown in Figure 6.6. The user can specify any number of criteria to search the database. Depending on the user input a query is generated. The query execution may retrieve a number of records. The listing of all these records is shown in the list box. The user can highlight a record to get a detail description of the record, which may help him in identifying the record. Once the record is identified, the user can select that record which then becomes the current record.

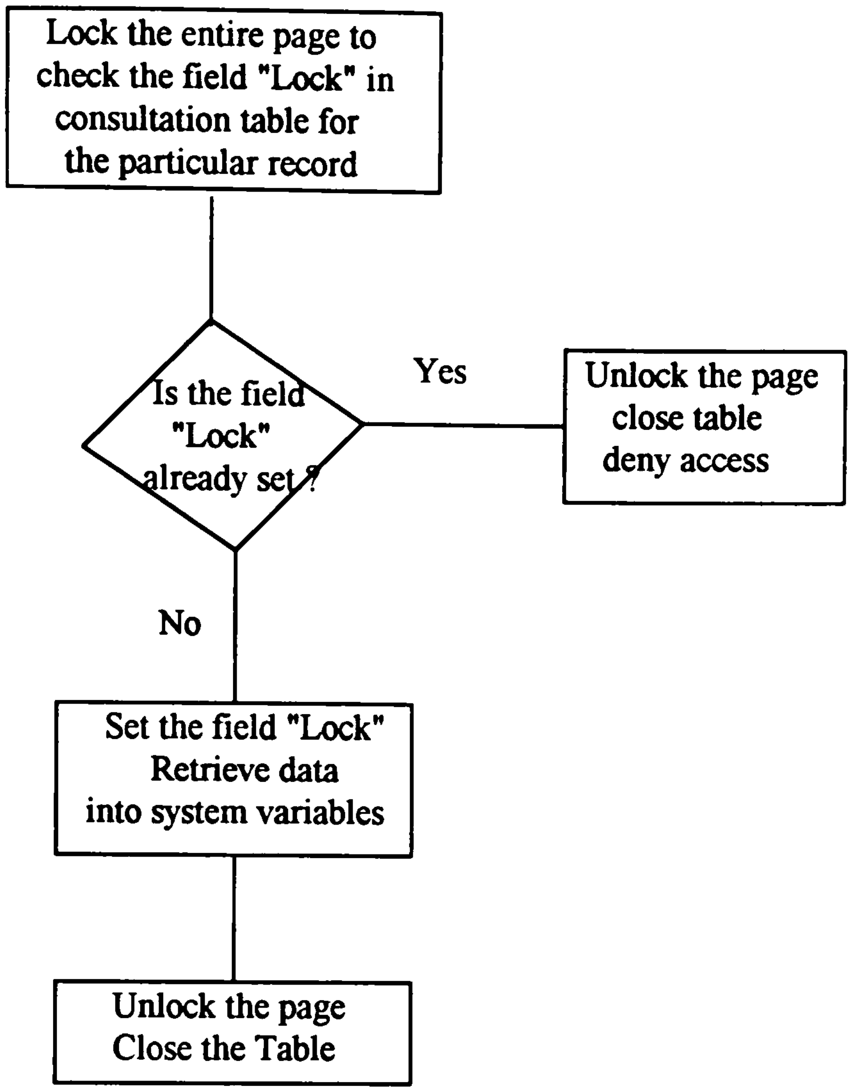

Lock the entire page to check the field "Lock" in consultation table for the particular record
Is the field "Lock" already set?
Yes
Unlock the page close table deny access
No
Set the field "Lock" Retrieve data into system variables
Unlock the page Close the Table

Figure 6.5 Locking Scheme

6.2.5 Update, Deletion and Addition of New Records

The users can update the records at any point. They can also retrieve the data, manipulate it, analyze it and still choose not to update it leaving the data unchanged. The locking scheme makes sure that the user has gotten the exclusive access to all of the data, and he is always looking at the latest data.

Before the user deletes the data permanently, a warning is given, upon confirmation the data is deleted. Before deleting a record, a user has to retrieve it. The record then naturally gets locked upon retrieval. Thus when the user deletes the record, he is the only one accessing it and other users do not become affected.

Adding a new record is handled in a similar way to updating a record. The user is prompted for a primary key for creation of new record.

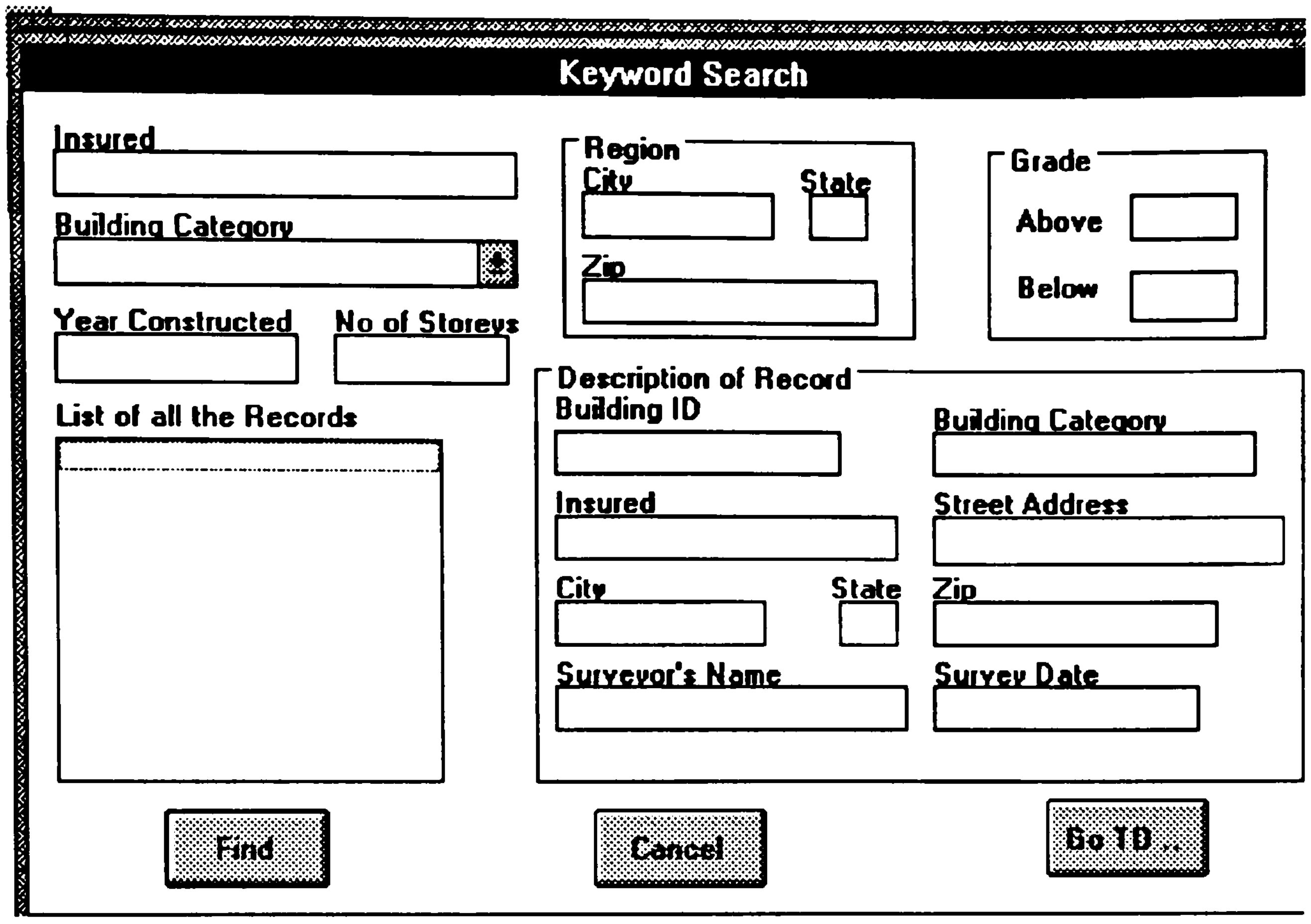


Figure 6.6 Retrieval Form

6.2.6 Maintaining the Speed of the Application

Database accesses can take considerable amount of time and such operations can reduce the application speed. For example, in the retrieval form when the user clicks on the consultation numbers in the list box, the system is expected to retrieve the details of that consultation number from two tables and display them on the screen. As the user action, clicking an element in the list box, does not take any time, the user expects the same kind of response from the system. There are a variety of places in the system where the user expects or demands good system response. It is known that the system response will largely depend on the machine the user is working on; however, the following things have been implemented to give users the best possible response on their machine.

1. Some faster database search command like "Seek" has been used. In Microsoft Access commands like "Move" or "Find" take comparatively more time than "Seek".

Wherever possible "Seek" has been used to get the best possible response on the available machine.

2. Some dynasets and tables are created as soon as the screen is loaded and they are destroyed only when the screen is unloaded. This reduces the overhead of creating, opening and closing of a dynaset each time the user triggers search or retrieve action.
3. Sometimes bookmarks are created which help the application remember the position in a table. Pointing back to a bookmark is much faster than searching that record all over again.

6.2.7 Unlock Utility

The locking scheme for this software has already been explained. It is possible that the system may crash when some user is working on it. It would mean that the system locked the record for the user, but it never unlocked it. After this, whenever users try to open that particular record, the system could give them a misleading message saying that the record is already locked and hence cannot be used. In order to avoid such a situation, an unlock utility has been implemented.

This unlock utility tries to obtain an exclusive access to the database to make sure that currently no other user is using the database. If no other user is using the database, then all the records should be in an unlocked state. The utility checks locks on each of the records and unlocks any locked records.

CHAPTER 7
FIELD SURVEY

7.1 Different User Groups

There should be a way to ensure that the system is built according to the specifications and the users find it satisfactory. It should also be found out if the system satisfies all the objectives and if it works efficiently and gives accurate results.
The intended group of users, experts and some general users, who are working on other expert system based applications, are the best people to judge this system. A survey was conducted in order to get feedback for the system. The users were classified as follows.

1. Experts and Knowledge Engineers: This group of users were the best people to judge if the system gave accurate results and if the domain knowledge provided and the improvements suggested were correct.
2. Beta testers from the Insurance Industry: The group of the beta testers from the insurance industry were the most appropriate to judge if the system met the requirements. They also judged if the system behaved accurately and consistently.
3. Actual users from the Insurance industry and other general users: This group consisted of actual or intended group of users from the Insurance Industry and also some general users who work on other expert system based applications. These people were capable of determining if the system was user friendly. They also determined if the system satisfied the user interface requirements of a general expert system based application. They gave their opinions on the database management aspect of this system.

7.2 Field Survey

The field survey was given to all the users and is presented in the Appendix.

7.3 Analysis of Replies

The analysis of all the responses given by the users is tabulated as below. The users were requested to give a rating for this software on the scale of 1 to 5, where 5 is the highest and 1 is the lowest rating.

Table 7.1 Analysis of Responses

Number of Participants : 25

	5	4	3	2	1	NA
User Interface	36	48	4	-	-	12
Database Features	52	44	-	-	-	4
Speed of Application	60	28	8	-	-	4
General Performance	36	44	-	-	-	20

(All values in tables are in percent)

CHAPTER 8
RESULTS AND CONCLUSIONS

This research has produced the following results and deliverables:

1. An integrated system with decision making and data management capabilities is ready for the insurance industry to evaluate and grade the buildings to be insured.
2. The aim of this thesis was also to provide a user interface which keeps the intended users in mind and meets their requirements. The aim was to meet the special requirements of user interface of expert system based applications.
3. The aim was to provide a basic outline and architecture for developing any other expert system based application with data management capabilities. For example, using the same kind of components, a system could be developed to diagnose patients and store the record of each patient.
4. The results of the survey was one of the deliverables of this research. This survey determined the overall effectiveness of this system. The results of this survey are stated in Chapter 7.

8.1 The Application

The software product resulted from this research can be very useful to the insurance industry. Some feedback has been received from the beta testers at the insurance industry. According to them, the software conforms to the requirements and hence can be immediately experimented with. This software will help the insurance industry in reducing paperwork and will provide them with an efficient information management tool. The software will also give consistent opinions for all buildings under consideration. Now the users will be able to query the data which was not possible earlier. The users can now obtain very useful domain and other information by accessing the help facility of this software. Other utilities which compact, repair and unlock the database will help users in maintaining it. Users can always convert the database records into readable text files and vice versa with conversion utilities.

8.2 System Analyses

One of the very important parts of this software cycle was system analyses. As pointed out by many software engineers and analysts, getting requirements from the customer is the most difficult part of the software cycle. This research was aimed at much more than just satisfying requirements. A detailed system analysis was done and alternatives were suggested to the customer. For example, the approach of database was suggested to the insurance industry. Many other features like graphical representation, statistical analyses, comparison of different buildings and database utilities were never demanded by the insurance industry. The aim of this research was to put on the user hat and think of all the features users may need now and in the near future. All of these additional features were thought of, after considering the current methods adopted by the insurance industry, by talking to other users who work on similar kind of software and by considering the features provided by the software platform.

8.3 Loosely Coupled Design

The more important aspects of the result of this research lie in the design of this software. As the design is loosely coupled, the components could be replaced individually without causing any changes in other components. For example, when the mode of information storage was changed from text files to database, the interface with the expert system component remained unchanged. There were no changes made in the way the expert system procedures were called and the way the DLL was interfaced. The user interface did not go through any major changes either except for the fact that some menus and screens were added to take care of interface with the database. All other stubs which dealt with the general aspects like presenting the screens, retrieving information remained unchanged. Most of the data structures remained unchanged, some more had to be introduced.

The design is open ended and hence future extensions can be done without a major overhaul in the system. More components could be added to the system in the same loosely coupled manner.

8.4 Usage of Object Oriented Concepts

The usage of some object-oriented concepts in the user interface improved the design very much. Earlier, each screen was represented by a different form in the design introducing over seventy forms in the project at design stage. This increased the project size immensely, consumed huge resources and resulted in a large executable file. The project took a long time to compile (about fifteen minutes) and make executable. The introduction of template forms changed all this. The size of the project reduced dramatically as the number of forms came down to two from seventy. The resources reduced, as the template form had only a command button, one label and a few three-dimensional command buttons at design time. The template form instantiated more of these resources at run time as and when needed. The height, width, alignment and overall appearance of this template form were calculated and adjusted at run time. Thus the alignment, the margins, allowances and overall appearances of all the instances were uniform at run time. The size of the executable reduced and the project compiling time came down by 75%.

There is some sacrifice in the speed of the application. This is more obvious on slower machines which take comparatively more time now to load forms. Once the forms are loaded the speed of the application is normal as usual. In order to encounter this problem, the forms are unloaded rarely. Once the forms are loaded for a particular database record, they are not unloaded until a different record is opened. The forms are repainted every time they are in focus.

8.5 Record Level Locking

Record level locking has been implemented in this application. Actually, the relational database management does not support record level locking. With the help of the relational database management, only page level locking could be implemented. Considering the amount of time for which the user holds on to the record, it was not feasible to lock the page for such a long time. The data is retrieved from the record into data structures. The user is actually manipulating or viewing the data in the data structures and not in the database. Whenever the user tries to update, the record is updated with the data structures. However in order to ensure that the user is looking at the latest data and in order to maintain consistency of the database, it is necessary to lock the database.

In order to implement locking, a new field was introduced in one of the main tables. This table, "consultation," keeps all the vital data regarding consultation. The moment the user retrieves data from a record, the field in the consultation table is marked as locked and is kept locked until the user closes the record. Before opening any record, the software checks for this lock and lets the user retrieve the record only if the record is marked unlocked. In this way a record level locking is implemented.

This type of approach facilitates multi-user access. As locking is on record level, the granularity is quite high. This approach can work for any other relational database. Introducing "Lock" fields in one or more tables can bring the effect of record level locking. The field can be a Boolean just carrying values as "Yes/No," saving a lot of space. However in such a case some unlocking utilities have to be written in order to unlock any permanently locked records in the event of a system crash. Unlocking utility for this particular record has already been written.

8.6 General Features

There are lot of other design features, which bring important results to the users. There are various functions written to assist users in carrying out statistical analysis on the data. Results are represented in graphical manner which help users understand the relative grading of the building under consideration. A user can compare several buildings depending on certain criteria. For example, a user can compare all buildings in New York City. The user is presented by a histogram which shows a general trend of buildings in

New York City. The user is also given some statistical facts like, mean and median of the grading, number of buildings and standard deviation.

8.7 Future Enhancements

As already stated, the system is open ended and hence implementing future enhancements should be relatively easy. A remote access to the database can be a worthwhile enhancement. All authorized users should be able to access the remote database of the insurance industry with read/write privileges. It would be an added advantage if the remote users could also do the statistical analysis and comparison.

It will be helpful to the users if some improvements are suggested to them regarding the building under consideration. After giving the grading, additional information regarding the improvements can broaden the applications of this software.

Implementation of additional graphics regarding the regional location can be an important future enhancement. This will give an exact idea to the user about tornado and hurricane regions in the country and if the building under consideration lies in the region.

LIST OF REFERENCES

1. Berry, Dianne & Hart, Anna. User Interface Standards for Expert Systems: Are they Appropriate?, Expert Systems with Applications, 1991, v 2 n 4, p. 245.

2. Christensen, Dawn M. The user interface: A Change in Communication, The Journal of Information Systems Management, Spring 1991, v 8 n 2, p. 71.

3. Di, Giorgi, Rosa, Maria, Fameli, Elio & Nannucci, Roberta. A Legal Expert System Prototype Integrated with Databases, Expert Systems with Applications, 1992, v 4 n 4, p. 397.

4. Elmasri Ramez & Navathe, Shamkant B. Fundamentals of Database Systems, 1989, The Benjamin/Cummings Publishing Company Inc., menlo park, CA.

5. Euler, Laura, Maffei, Eric & Rauch, Adam. Create a Real Windows Applications in a Graphical Environment Using Microsoft Visual Basic, Microsoft Systems Journal, July 1991, v 6 n 4, p. 57.

6. Ghosh, D. K. & Kalyanraman, V. A Relational Database Interface for Design Expert Systems, Microcomputers in Civil Engineering, June 1990, v 5 n 2, p. 151.

7. Horward, H. Craig & Rehak, Daniel R. KADBASE: Interfacing Expert Systems with Database, IEEE Expert, Fall 1989, v 4 n3, p. 65.

8. Hendler, James A. Expert Systems-The User Interface, 1988, Ablex Publishing Corporation, Norwood, NJ.

9. Kakashima, Yusei, Daito, Noriaki & Fujita, Satoru. Integrated Expert System with Object-Oriented Database Management System. Systems and Computers in Japan, 1992, v 23 n 11, p. 29.

10. Kerschberg, Larry. Expert Database Systems: Knowledge/Data Management Environments for Intelligent Information Systems, Information Systems, 1990, v 15 n 1, p. 151.

11. King, Dave. Intelligent Decision Support: Strategies for Integrating Decision Support, Database Management, and Expert System Technologies. Expert Systems with Applications, 1990, v 1 n1, p. 23.

12. Miller, G. S. & Colton, J. S. The Complementary Roles of Expert systems and Database Management Systems in a Design for Manufacture Environment. Engineering with Computers, Summer 1992, v 8 n 3, p. 139.

13. Perez, R. A. & Koh, S. W. Integrating Expert Systems with a Relational Database in Semiconductor Manufacturing, IEEE Transactions on Semiconductor Manufaturing, Aug. 1993, v 6 n3, p.199.

14. Risch, Tore, Reboh, Rene & Hart, Peter. A Functional Approach to Integrating Database and Expert Systems, Communications of the ACM, Dec. 1988, v 31 n 12, p. 1424.

15. Ruiu, D., Divi, R. & Katzberg, P. Expert System and Relational Database Improve Planning Productivity, IEEE Computer Applications in Power, July 1992, v 5 n 3, p. 39.

16. Rumpel D. & Krost, G. Natural language Interface and Darabase Issues in Applying Experts Systems to Power Systems, Proceedings of the IEEE, May 1992, v 80 n 5, p. 758.

17. Sakr, K.M. & Hosain, M. U. Applications of Expert System Building Tools in Structural Design. Canadian Journal of Civil Engineering, June 1991, v 18 n 3, p. 493.

18. Schon, Stephen & Helferich, Omar Keith. Expert System Applications in Customer Service, Data base, Summer 1989, v 20 n 2, p. 38.

19. Segev, Arie & Zhao, Leon. Rule Management in Expert Database Systems, Management Science, June 1994, v 40 n 6, p. 685.

20. Wexelblat, Richard L. On interface requirements for Expert systems, AI magazine, Fall 1989, v 10 n 3, p. 66.

21. Whang, Kyu-Yong & Navathe, Shamkant B. Integration of Expert systems and Database Management Systems-An extended disjunctive normal form approach, Information Sciences, Oct. 1992, v 64 n 1, p. 57.

22. A Report: New Directions in Human computer Interaction: Education, Research and Practice. 1994, National Science Foundation and Advanced Research Project Agency, Washington, D.C.

APPENDIX

FIELD SURVEY

Field Survey

Occupation: Student ( ) Other ( ) ______________________________

Name of the University/Organization: ______________________________

The computer system on which the system was tried out was a

386 SX DX	486 SX DX	Pentium	Other: ______________

Introduction:	Yes	No
1. Do you use other windows software frequently? If <u>Yes</u> list the couple of frequently used software: ______________________________	( )	( )
2. Are you an expert on the subject matter? (Expert here means having knowledge about wind hazards to structures)	( )	( )

General Information:	True	False
1. This is a expert system based software to grade buildings for wind hazard	( )	( )
2. This is a expert system based software to grade buildings for wind hazard and other natural hazards	( )	( )
3. This system has integrated both databases and expert system technologies	( )	( )

General system operation:	Yes	No
1. Were you able to install the software without any difficulties?	( )	( )
2. Were you able to run the software without getting any system errors? (For example, General Protection Faults)	( )	( )

Comments:

User Interface Yes No

1.	Was it easy to navigate through the system?	( )	( )
2.	Were you just stuck sometimes not knowing what to do next?	( )	( )
3.	Was the relative speed of operation of this system comparable to other average windows software running on your system?	( )	( )
4.*	Were the questions and choices easy to interpret? (For people with knowledge of subject matter only)	( )	( )
5.	Were the menu titles, button captions and other text in the system self explanatory ?		

Comments:

____

____

* To be answered only by experts on the subject matter

Database:		Yes	No
1.	Have you worked on any other database based software? If Yes list some of them ______________________	( )	( )
2.	Were you able to open the database and use other database features without any difficulty? If No list the problems encountered: ______________________	( )	( )
3.	Was retrieving records easy and convenient?	( )	( )
4.	Was the speed of database access and update satisfactory?	( )	( )

Expert System Features		Yes	No
1.	Have you used any other expert system based software?	( )	( )
2.	Were you able to carry out a successful consultation? If No state the problem encountered: ______________________	( )	( )
3.	Was it easy to interpret the results?	( )	( )

Section for Subject Matter Experts

1.*	Does this system give correct Results?	( )	( )
2.*	Are there any other tools to validate the results of this system	( )	( )
3.*	Has the system been validated using those tools?	( )	( )
4.*	Are the results obtained from the tool and this system matching?	( )	( )

* To be answered only by experts on the subject matter

General Performance*		1	2	3	4	5
1.	How is the performance of this system compared to the other expert system based software?	( )	( )	( )	( )	( )
2.	How is the user interface of this system compared to other expert system based software?	( )	( )	( )	( )	( )
3.	How is the data storage and management capabilities of this system compared to the other expert system based software?	( )	( )	( )	( )	( )
4.	How is the system performance and speed compared to other windows software?	( )	( )	( )	( )	( )
5.	How is the user interface and retrieval process of this system compared to other database based software?	( )	( )	( )	( )	( )

* Rank the general performance on the scale of 1 to 5, where 5 corresponds to the highest and 1 corresponds to the lowest general performance.

General Comments:

____

____