dedicated to
my sister
Engin

AN EXPERT SYSTEM APPLICATION

by

GÜLSELİ ZEREN

B.S. in I.E., Bogaziçi University, 1985

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Industrial Engineering

Boğaziçi University

1987

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Doç. Dr. M. Akif Eyler whose motivation and inspiration led me to finish this study. I should say that anything good in this thesis is due to his guidence.

I also would like to express my gratitude to Doç. Dr. Selahattin Kuru and Prof. Dr. İbrahim Kavrakoğlu for their valuable advice and comments.

In particular, I wish to thank my sister, Doç. Dr. Nuran Zeren, who helped and supported me as being an experienced expert on conservation decision making.

Finally, of course, I must thank to my mother and father for tolerating my sufferings throughout this study.

AN EXPERT SYSTEM APPLICATION

ABSTRACT

Expert Systems deal with difficult, ill-structured problems in complex domains for which no straight forward algorithmic solutions exist. Mostly these problems need human experts, with years of training and education, specialized in the area of concern. However it may not be always possible to have an access to these scarce and valuable experts. It is at this point where the importance of Expert Systems becomes clear. Expert Systems are designed to solve problems more or like in a similar manner as real experts. They are defined to be systems aiming at assisting in problem analysis and decision making.

One of the complex areas, which is faced with scarce expert problem is the conservation decisions that should be given over countless cultural property in Turkey. This thesis suggests an Expert System in order to clarify the conservation type and conservation degree depending on the characteristics of the cultural property. As a consequence of this clarification the conservation rationale, which is very hard to determine, will be established; so that the cultural properties will have the chance of inheriting their quality and physical conditions over the years.

UZMAN SİSTEM UYGULAMASI

ÖZET

Uzman Sistemler karmaşık alanlardaki, zor ve bozuk yapılı, algoritmik çözümleri olmayan problemlerle ilgilenir. Esasta bu tür problemlerin çözümünde, bu alanda yıllar boyu eğitim ve öğretim görmüş uzmanlara gereksinim vardır. Bununla birlikte, çok değerli ve sayıları az olan bu tür uzmanlara her zaman ulaşabilmek mümkün olmaz. İşte bu noktada Uzman Sistemlerin önemi açıkça ortaya çıkar. Uzman Sistemler problemleri gerçek uzmanların yöntemleri gibi ya da benzer şekilde çözebilmek için tasarlanmıştır. Bu sistemlerin problem incelemede ve karar verebilmede yardımcı olması amaçlanmıştır.

Türkiye'deki sayısız kültürel yapı için verilmesi gerekli koruma kararları sınırlı uzman sorunu olan karmaşık alanlardan biridir. Bu tezde, kültürel değerlerin özelliklerine uygun koruma yöntemi ve koruma derecesini aydınlatmayı amaçlayan bir Uzman Sistem önerilmektedir. Bu sistemin kullanımı ile tanımlanması çok zor olan koruma mantığı yerleşecek ve böylece kültürel değerler özelliklerini yıllar ötesine taşıma şansına sahip olacaklardır.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

In modern business life, everyone is faced with complex problem solving and decision making based on extensive but incomplete, uncertain and even contradictory data and knowledge. Providing solution for these less formalized or understood problem areas by using conventional programming techniques is nearly impossible. This restricts the use of computers in certain problem areas. The desire to use computers in solving these less formalized problems leads into the recent interest in Expert Systems.

Expert Systems enlist the computer in a healthy and powerful way to solve difficult and important problems. A 'proper' Expert System can be far more useful and reliable than any expert information source otherwise available. These characteristics made Expert Systems a challenging topic for investigation in this study.

Through out the study, mainly three activities are emphasized :

- To develop a simple Expert System program (shell) with its inference mechanism, knowledge acquisition or learning, and explanation components

- To apply a specific knowledge domain onto this program and observe the results

- To compare the results obtained with the domain expert's decisions on some case applications

Related to these activities, following topics are covered in this thesis.

1- What Expert Systems are and How they are set up

2- How the new Expert System program is developed

3- What the knowledge domain is and Why it is chosen

4- How the knowledge base developed responds to real life situations

In the first part, general descriptions about Expert Systems are summarized while stressing the associated pros and cones. In the second part, the characteristics related to the new program developed are described in coordination with the general descriptions made in the first part. In the third part, characteristics of the knowledge domain are introduced. In the last part, domain expert's reactions on the system developed and the results obtained by running the knowledge base on 14 different cases along the Tarabya - Yeniköy Coast are presented thus supplying some kind of performance measure for the new program.

II. GENERAL OVERVIEW ON EXPERT SYSTEMS

2.1. EVOLUTION AND DEVELOPMENT OF INTERACTIVE APPLICATIONS

With the introduction of the computer as a powerful tool to humanity, all attention was given to developing softwares to aid in solving problems. These softwares required an explicit formalization of the problem into detailed sequential statements before providing a solution. Therefore, the softwares written by these conventional programming techniques were complex and sophisticated in nature [14].

Around 1960's the development methodology was to make isolated application programs by using these type of softwares. Each application needed different programs and as a further consequence, changes extensions to the problem solving knowledge required extensive and painful program maintenance work [4]. All knowledge about an application was in terms of programs and files where these programs and files were embedded and optimized upon each other [2]. The programmer in writing the software had to assure 'completeness', that is the program had to provide actions for all possible combinations of conditions, 'uniqueness', that is the output had to be unique for a certain set of conditions, and 'correctness', that is the set of rules had to provide a correct outcome for all possible conditions . This development methodology was data and rule dependent [23].

In 1970's Database System type of methodology was evolved where the factual knowledge was in the form of a database; therefore the

application programs turned out to be data independent. This methodology solved the problem of data dependence however, 'rules' for processing still stayed embedded in application programs [2].

Even though Database Systems were data independent, rule dependence made them impossible to automate less formalized problems. However, through the advancements in Artificial Intelligence and subsequent emergence of Expert Systems, Database Systems later led into Knowledge Base Systems where factual knowledge was again in database like Database Systems but in addition, rule knowledge was in knowledge base. With this new technology program code turned out to be application independent which made it possible to develop quick and pragmatic answers for a wide range of problems that currently defy effective solutions [6]. Table 1 summarizes the characteristics of different application systems.

TYPE	RULE	DATA	RESULT
1- Conventional Programming Techniques	Dependent	Dependent	Very Inflexible
2- Database Systems	Dependent	Independent	Inflexible
3- Knowledge Base Expert Systems	Independent	Independent	Flexible

TABLE 1 : Characteristics of Different Application Systems

2.2. APPLICATION AREAS OF EXPERT SYSTEMS

Expert Systems deal with difficult, ill-structured problems in complex domains for which no straight forward algorithmic solutions exist [23]. The range of potential Expert System applications covers a spectrum from derivation problems to formation problems [19].

In derivation problems, the problem conditions are specified as parts of a solution description (goal). This means that the possible outcomes exist in the knowledge base. Expert Systems for derivational problems try to apply the available knowledge and rules such that the initial data and conditions are well integrated in the solution. In formation problems, conditions are given in the form of properties that the solution as a whole must satisfy. Candidate solutions are generated and tested against the specified constraints.

Most actual problems can not be classified as purely derivation or formation problems; but they lie somewhere in between. The actual problems which cover the spectrum between the two problem types can be seperated into mainly eight different systems:

- Interpretation Systems
- Prediction Systems
- Diagnosis Systems
- Monitoring Systems
- Design Systems
- Planning Systems
- Repair Systems
- Control Systems

Interpretation Systems take the observed data and explain its meaning by inferring the problem state which corresponds to the observed data [19].

Prediction Systems infer likely consequences starting with a given situation [23].

Diagnosis Systems infer malfunctions from observed irregularities and interpretation of data [19].

Monitoring Systems observe the system behavior and compare the observations to the planned behavior to determine flaws in the plan or potential malfunctions of the system [19].

Design Systems develop a configuration for an object which satisfies applicable constraints [19].

Planning Systems try to set up a program of actions to achieve certain goals. In doing the set-up, this system should not exceed the resources and violate the constraints [6].

Repair Systems plan remedies for malfunctions found through diagnosis.

Control Systems encompass many of the caracteristics of the system described. They must interpret data, predict outcomes, formulate plans, execute plans and monitor execution [19,6].

Interpretation, Prediction, Diagnosis and Monitoring lie at the derivation end of the spectrum while Design, Planning Repair and Control lie at the formation end [23].

2.3. ARCHITECTURE OF EXPERT SYSTEMS

Expert Systems are computer programs that are designed to solve problems in a specific domain of knowledge in a manner similiar to that of experts with years of training and education [4]. The architecture of Expert Systems is such that they are composed of six parts :

- User Interface or Dialog Component
- Knowledge Base Component
- Inference Mechanism or Deduction Component
- Context or Working Memory
- Explanation Facility
- Knowledge Acquisition Component

These six parts are briefly defined in the following paragraphs and the interactions between these parts are presented in Figure 1.

1- User Interface or Dialog Component

The system is accessed through an interface which provides a link between the user and the Expert System. The link mechanism is responsible for controlling and translating the user specified input into a form acceptable by the system as well as for the output presented to the user [14]. Unfortunately, there is no perfect human-computer interface. Proper interface should depend on the application environment, expertise, interest, variation among users, software performance, recent experience [12]. Most important of all user interface for different tasks should be consistent all through the program.

2- Knowledge Base

Knowledge base contains a collection of general facts, rules and heuristic knowledge for a particular application domain [15]. A number of formalisms, frames, logic and semantic nets can be used to represent knowledge.

3- Inference Mechanism or Deduction Component

The inference mechanism controls the processing of the program by using the knowledge base to deduce new facts which can be used for subsequent inferences [15]. The inference mechanism operates on the context or working memory of the system [19]. Its objective is to arrive at a global conclusion. The process continues until the problem is solved or when there are no more rules remain to be processed.

4- Context or Working Memory

Context contains all the information which describes the problem currently being solved, including both problem data and solution status. The problem data can be divided into facts provided by the user and those derived or inferred by the program. Context has dynamic structure which exists only during consultation sessions.

5- Explanation Facility

Explanation facility of the Expert System provides the reasoning and problem solving strategy to the user. In the consultation stage the user may interrupt the system and inquire what is being done and why the current line of reasoning is being pursued. In addition, the program can explain, in an a-postriori fashion, how any fact was deduced and how knowledge was applied [19].

6- Knowledge Acquisition Component

The information in the knowledge base is usually in a complex and rigid format. The translation of knowledge obtained from expert(s) to the required format may be tedious [19]. The knowledge engineers may fail to fully understand the nature of the assistance needed by the end user [24]. Knowledge acquisition is, therefore, introduced to provide means for entering and revising knowledge easily in the knowledge base [14].

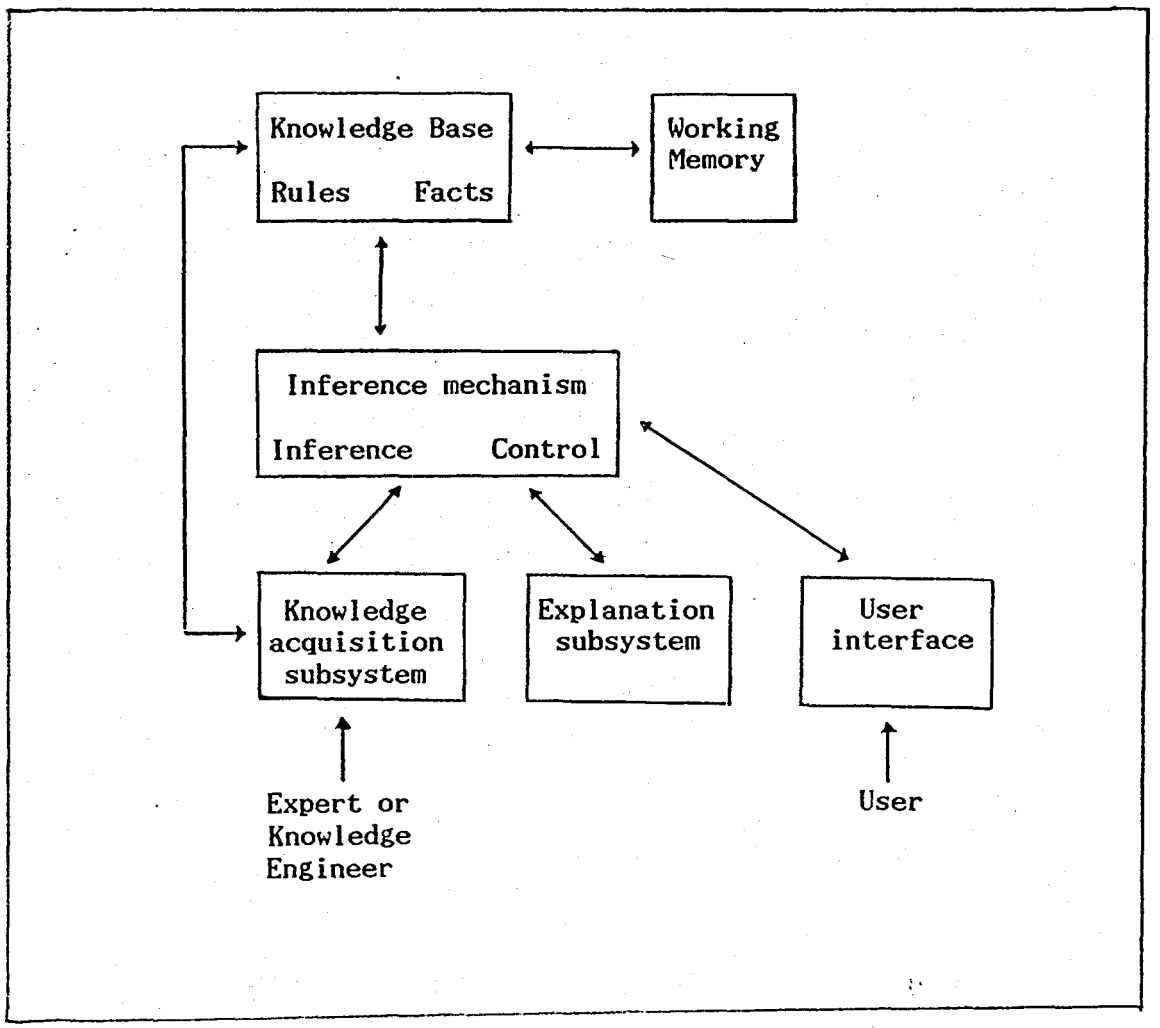

FIGURE 1 : Architecture of Expert Systems [6]

2.4. DEVELOPMENT OF EXPERT SYSTEMS

Development of an Expert System involves a team effort in which mostly domain experts (DE) supply the knowledge on the problem area and knowledge engineers (KE) acquire this knowledge and embed it in Expert Systems. Therefore, DE and KE are the key elements in the development of Expert Systems who should work collectively as a team all through [19,6] Of course, there may be some other people like management and users involved in developing Expert Systems; first being the media for identifying the need and second being the media for actually using the system. In Figure 2 different characteristics of people involved in Expert Systems are summarized.

Expert System development starts with the selection of appropriate language or framework [19]. This means that definitions of inference mechanism, knowledge base and context structure should be completed before proceeding any further [6].

Second step is identifying the important aspects and characteristics of the problem. This stage mostly requires two distinct people one being the domain expert (DE) who supplies the problem solving knowledge for the problem area , and the other being the knowledge engineer (KE) who gathers expertise from the DE and translates this knowledge into the format required by the system.

Third step is formalizing the concepts identified in the previous step. Initially this involves describing the system on paper and matching the concepts with formal representation tools and schemes defined in the first stage [23].

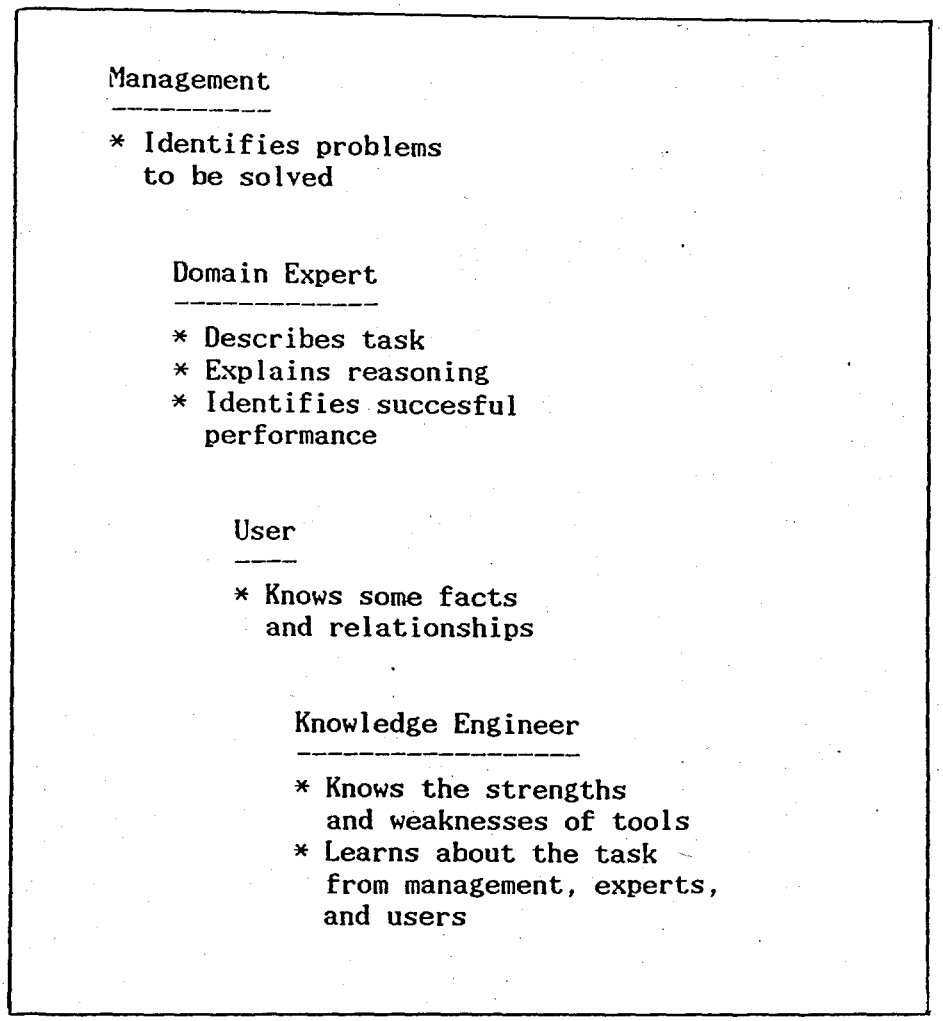

FIGURE 2 : Different Roles in Expert System Development [6]

Fourth step is implementing the knowledge obtained. KE does the encoding in the chosen tool. Initially a prototype system is developed where the knowledge base consists of the KE's understanding of the problem [23].

Fifth step is testing the prototype system developed. This is done by the DE. Weaknesses and mistakes both in knowledge base and inference mechanism are identified [6]. Mostly due to the disagreement

and inconsistencies detected, the system will be revised where formalization or implementation operations will be repeated again.

Last step is the Maintaining and Updating the system. After this step Expert System development finishes. These steps are summarized in Figure 3.

Steps in Development	Responsible Person
1 - Select Tools	KE
2 - Identify Problem	KE & DE
3 - Design System	KE
4 - Develop Prototype	KE
5 - Test & Revise	DE & KE
6 - Maintain & Update	DE & KE

FIGURE 3 : Development of Expert Systems

2.5. PROBLEM SOLVING STRATEGIES FOR EXPERT SYSTEMS

Problem solving in Expert Systems involves the search for a solution through a state-space by the application of operators, where the state-space (the possible states in the problem solution) consists of an initial state, a goal state and intermediate states. A solution path consists of all states that lead from the initial state to the goal

state [19]. There are a number of different problem solving strategies used in current Expert Systems which is described briefly.

-Forward Chaining : A system is said to exhibit forward chaining if it works from an initial state of known facts to a goal state [15]. In this strategy all facts are input to the system and the system deduces the most appropriate goal state that fits the facts.

-Backward Chaining : A system is said to exhibit backward chaining if it tries to support a goal state by checking known facts in the context [15].

-Mixed Initiative : A system uses a mixed initiative startegy when it combines the forward and backward chaining [13].

-Means-ends Analysis : In means-ends analysis, the difference between the current state and the goal state is determined and used to find an operator most relevant to reducing this difference [15].

-Problem Reduction : Problem reduction involves factoring problems into smaller subproblems. The problem is presented as an AND-OR graph [13].

-Plan-Generate-Test : In this strategy, all possible solutions in the search space are generated and each solution is tested until a solution that satisfies the goal condition.

-Backtracking : In backtracking the problem solver backs up to a previous level in the solution process if no solution is found in the current path [13].

-Hierarchical Planning & Least Commitment Principle : Hierarchical planning involves developing a plan at successive levels of abstraction where these levels are loosely coupled [13]. The least commitment principle involves defering the assignment of values to variables until more information about the problem space is available [19].

-Constraint Handling : Constraint satisfaction method involves the determination of problem states that satisfy a given set of constraints [15].

-Agenda Control : The agenda control strategy involves assigning a priority rating to each task in the agenda. The task with the highest priority is performed first [13].

2.6. LANGUAGES OR TOOLS FOR EXPERT SYSTEMS

Expert System applicability will broaden in time with the appropriate choice of tools for the purpose [8]. In the development phase of Expert Systems, therefore, detailed considerations have to be given for selecting the appropriate language, environment, or tool. The media available for developing Expert Systems can be analyzed as General Purpose Programming Languages, General Purpose Representation Languages and Domain Independent Expert System frameworks (Shells).

Expert System developers commonly use high-level languages to implement projects. These high-level languages contain some special features, such as facilities for handling large chunks of knowledge and operators for developing, planning and reasoning [15]. These languages have powerful abstraction mechanism with which other high-level

constructs can be built so as to make programming feasible and easy [19]. Commonly used languages like BASIC, FORTRAN or PASCAL can also be used as a development language but such languages are far from ideal in representing the real-world knowledge [22]. Among many of the high-level languages LISP and PROLOG are the popular ones.

LISP which stands for LISt Processing language is designed so that there is no essential difference between data and programs. This means that LISP programs can use other LISP programs as data. LISP is highly recursive, and data and programs are both represented as lists [6].

PROLOG, which stands for PROgramming in LOGic [25], is designed for symbolic rather than simply numerical computation. PROLOG is very efficient in list processing and can respond to any query by attempting to return an answer immediately since it is an interpreted language. PROLOG programmer does not specify how the computer is to perform its task but rather specify the description of the task as a sequence of constraints to be satisfied [6].

General Purpose Representation Languages are languages developed specially for knowledge engineering [15]. These languages are not restricted to implement any particular control strategy, but facilitate the implementation of wide range of problems spanning the derivation-formation spectrum [19]. Some General Purpose Representation Languages are : SLR, RLL, KEE, OPS5, ROSIE, LOOPS, AGE [19].

Domain Independent Expert System Frameworks are designed to facilitate the rapid development of Knowledge Systems. They incorporate specific strategies for representation, inference and control. Some

examples for these type of frameworks are EMYCIN, KAS, HEARSAY-III, EXPERT, KES [15] .

2.7. PROBLEMS ASSOCIATED WITH EXPERT SYSTEMS

In all the previous sections the benefits that result from Expert System applications are mentioned. (These benefits are summarized in Table 2). In a research made over nine countries in four continents about 95% of the respondents said that they saw Expert Systems as very important or vital to their business [20]. On the other hand, it is worthwhile to mention the problems associated with planning, developing, implementing and using of Expert Systems.

* Heuristics
* Highly Interactive Processing
* Replication of Human Behavior
* Symbolic Processing
  (symbolically structured knowledge base in a global working memory)
* Mid-Run Explanation
* Decision Making
* Serving Different types of Users
* Handling Unanticipated Input

TABLE 2 : Benefits of Using Expert Systems [6,23]

Problems associated with Expert Systems may be grouped into seven. Problems in each group will be presented as only items since the discussion of each problem is a large topic by itself.

1- Difficulties in planning an Expert System [10]

- Choosing the right application and cutting it down to size (ie. reducing its scope)
- Putting (potential) user expectations into the right perspective
- Obtaining full commitment from within the organization and thus the necessary resources
- Choosing appropriate tools for developing such systems

2- Problems encountered in obtaining expertise [10]

- Lack of written source of knowledge
- Contradicting knowledge of different experts
- Commitment of the expert(s) to the project
- Difficulty in representing the actual knowledge in a suitable form for the logical processing of the computer

3- Difficulties in developing and testing [10]

- Controlling the size of the problem
- Estimating the development time and resources required
- Having incremental and interactive approach (need of close interaction from the expert and the user)
- Providing a good explanation facility

4-Problems in getting the Expert System accepted by the users [10]

- High level of user expectations
- System seen as 'threat' rather than 'help'
- Liability of the system
- System performance and reliability

5- Problems holding back the widespread application of Expert Systems in commercial environment [10]

- Expert Systems are 'new' commercial technology
- There is a lack of staff skilled in building such systems and the number of users familiar with such technology is small
- Most of the current Expert Systems are based on the use of specialized hardware and software
- There is a lack of integration between Expert Systems and 'conventional' Data Processing Systems

III. NEW EXPERT SYSTEM PROGRAM

The new Expert System application contains all the six components of a typical Expert System : user interface, knowledge base, context, inference mechanism, explanation facility, and knowledge acquisition components. Turbo PROLOG version 1.1 is used in developing the programs for user interface, inference mechanism, explanation facility, and knowledge acquisition. All the six components reside in program ARCH.PRO Detailed explanations on each component are given in the following sections.

3.1. USER INTERFACE

User interface in this study, is a menu-driven system aiming at providing a smooth interaction between the user and the system. This interface contains some control remarks in order to prevent incorrect decisions of the user. The user has the option of loading, consulting, saving and acquiring knowledge which will be selected by using a menu. A program also developed in PROLOG is used to run this menu-driven system: MENU.PRO. All through the execution this is used to get information from the user.

3.2. KNOWLEDGE BASE

In this study, the knowledge base is made up of five different

database relations. First one is for representing the empirical associations or rules. The second one is for representing the factual knowledge. The third and fourth ones are for representing logical relations and characteristics respectively. The fifth one is for keeping the state-space domains.

The emprical associations are organized into levels of abstractions. These abstractions are hierarchical levels of the knowledge where different levels are connected in a tree-like structure (Figure 4) forming classes and subclasses.

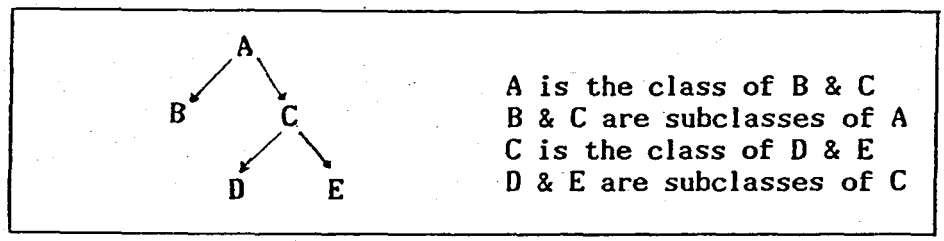

FIGURE 4 : Hierarchical Levels in a Tree-Structure

Each node in the structure can be represented by its class name, its own subclass name, and the relation between itself and the class. In this study database to represent each node is the Rule Database. It consists of three parts to describe three sets of knowledge explained early in the paragraph.

Rule Database Representation : f(class,subclass,list)

where class contains the name of the class

subclass contains the name of the subclass

list contains the list of conditions to satisfy relation

Consider a simple tree like in Figure B and Table A, and suppose that an animal is a penguin.

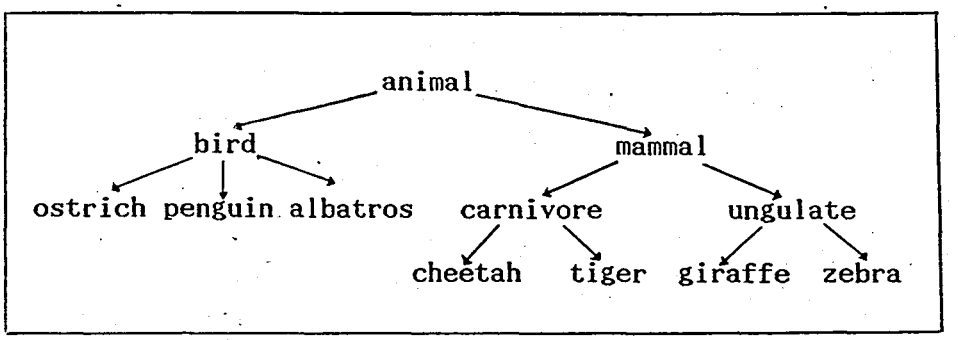

FIGURE 5 : Representation of a Simple 'Animal-Tree'

Relation	Condition
Animal to Bird	has feathers
	lays eggs
Animal to Mammal	has hair
	gives milk
Bird to Ostrich	can fly
	has long legs
	has long neck
	has black and white color
Bird to Penguin	can swim
	not can fly
	has black and white color
Bird to Albatros	can fly
	not has long legs
Mammal to Carnivore	eats meat
Mammal to Ungulate	not eats meat
	chew cud
Carnivore to Cheetah	has tawny color
	has dark spots
Carnivore to Tiger	has tawny color
	has black stripes
Ungulate to Giraffe	has long legs
	has long neck
	has dark spots
Ungulate to Zebra	has black stripes

TABLE 3 : Relations and Conditions in 'Animal-Tree'

In the animal-tree in order to reach from animal node to penguin node, bird node has to be reached first. So for an animal to be a penguin that animal should have all the conditions to satisfy the animal to bird relation plus the bird to animal relation.

```
====> An animal should - have feathers
                       - lay eggs
                       - swim
                       - not fly
                       - have black and white color
```

Rule Database for this example will contain two different entries;

one for Animal to Bird Relation

one for Bird to Penguin Relation

First two parts of each database is easy to form, but it is very hard to represent conditions as phrases since they take too much time and space. To make things easier, these phrases are collected in a different database, Fact Database in which they are given an index. This whole thing is called as a fact. Full Fact Database related to the 'animal' problem is given in Table 4-a.

Fact Database Representation : fact(no,phrase)

where no is the index of the fact

phrase is the actual wording of the fact

After the introduction of Fact Database each condition in the Rule Database can be represented by using the index of the fact.

A phrase can be negative or positive in the condition list.

EX: phrase - can fly
condition - not can fly

In order to differentiate conditions in such a situation index number is multiplied by (-1) to make the condition negative of the phrase. So the absolute value of the number in the condition list represents the fact index, sign of the number represents the Negative or Positive Fact Relation. Rule Database for the 'animal' problem is given in Table 4-b.

(a)
Fact Database

```
fact(1,"has feathers")
fact(2,"has hair")
fact(3,"lays eggs")
fact(4,"gives milk")
fact(5,"can fly")
fact(6,"has long legs")
fact(7,"has long neck")
fact(8,"has black and white color")
fact(9,"can swim")
fact(10,"eats meat")
fact(11,"chews cud")
fact(12,"has tawny color")
fact(13,"has dark spots")
fact(14,"has black stripes")
```

(b)
Rule Database

```
f("animal","bird",[1,3])
f("animal","mammal",[2,4])
f("bird","ostrich",[5,6,7,8])
f("bird","penguin",[9,-5,8])
f("bird","albatros",[5,7])
f("mammal","carnivore",[10])
f("mammal","ungulate",[-10,11])
f("carnivore","cheetah",[12,13])
f("carnivore","tiger",[12,14])
f("ungulate","giraffe",[6,7,13])
f("ungulate","zebra",[14])
```

TABLE 4 : Fact and Rule Databases for 'Animal-Tree'

In real life, conditions can have logical relations among each other; one fact can imply others.

Consider another very simple tree as in Figure 6.

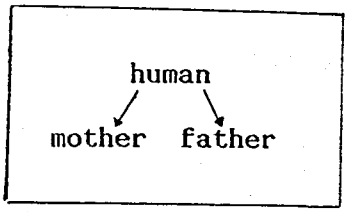

FIGURE 6 : Representation of a Simple 'Human-Tree'

Relation	Condition
Human to Father	is a male
	has a child
	is a parent
Human to Mother	is a female
	has a child
	is a parent

TABLE 5 : Relations and Conditions of 'Human-Tree'

(a) Fact Database	(b) Rule Database
fact(1,"is a male")	f("human","father",[1,3,4])
fact(2,"is a female")	f("human","mother",[2,3,4])
fact(3,"has a child")	
fact(4,"is a parent")	

TABLE 6 : Fact and Rule Databases for 'Human-Tree'

According to what has been said until now the representation of the human problem in Figure 6 and Table 5 is represented in Table 6. Apart from these relations there may be other logical relations among conditions. Examples of such relations can be detected from Table 6. If a human is known to be a non-male then this means, that human is a

female. In this situation there is no need for trying to satify the condition "is a female". Again if a human is known to have a child this means, that human is a parent. Therefore there is no need for trying to satisfy the condition "is a parent". Some other same type of examples are summarized in Table 6.

Fact	implies	Fact
"is a male"	<----->	Not"is a female"
"is a female"	<----->	Not"is a male"
"has a child"	<----->	"is a parent"

TABLE 7 : Logical Relations in 'Human-Tree'

In order to represent these type of logical relations another database namely Logical Relations Database is introduced. Table G.

Logical Relations Database Representation : implies(no1,no2)

where , absolute value of no1 represents the implying fact index
sign of no1 represents the negative or positive implication
absolute value of no2 represents the implied fact index
sign of no2 represents the negative or positive implication

```
implies(1,-2)
implies(2,-1)
implies(3,4)
implies(4,3)
```

TABLE 8 : Logical Relations Database for 'Human-Tree'

As a characteristic of real life applications some conditions can only be implied. These type of conditions are mostly subjective facts which most of the time depend on other facts, they can only exist objectively when inferred by some objective facts. There is no sense in asking about a subjective fact to the user since the answer will depend on the evaluation of the user. In such cases domain expert's decision criteria should be used to make the particular fact an objective one. The objective fact together with the facts that imply it asserted in Logical Characteristics Database.

Logical Characteristics Database: implied(no,list)

where no is the index number for the fact which should be implied

list is the list of conditions that implies the fact

Consider an island : One fact about this island could be that it is small. This fact is really subjective; a user from Australia could name it as small but a user from Philiphines as large. The problem here is the un-defined decision criteria. In this case fact "is small" should be implied by other facts, one of which could be the area of the island. If the domain expert in this area defines the small island as "an island less than ten square kilometers" than the implying fact becomes the "is less than ten square kilometers". So, before deciding on whether an island is small or not the actual square kilometers has to be known. Depending on it the answer island can be named as small or large.

The knowledge base with the previously defined four Databases is totally independent of the program. This independence is the main logic behind Expert Systems however there must be some kind of

connection between knowledge base and the program in order to have a smooth consultation mechanism. The connection to make the program be aware of the knowledge base is achieved by introducing another database for state-space domains. This database named the Type Database contains all the possible classes in the knowledge base.

Type Database : type(class)

where class is the name of possible state-space domain

As a summary the knowledge base of this study contains four different Databases :

1- Rule Database : f(class,subclass,list)

2- Fact Database : fact(no,phrase)

3- Logical Relations Database : implies(no,no)

4- Logical Characteristics Database : implied(no)

5- Type Database : type(class)

3.3. CONTEXT

Knowledge base contains the static knowledge of the problem domain. During consultation some information will be generated on a particular program execution, reflecting the current state of the problem. This information which is dynamic in sturucture and which exists only during the execution stage, forms the context part of the Expert System. In this study, there are seven different relations to represent the dynamic knowledge in the context.

- Candidate Database

This database has the list of conditions in the candidate solution. Due to the nature of programming language used (Prolog), it is impossible to keep the initial condition list as a whole during execution. For this reason the initial list is asserted to Candidate Database which makes it possible to obtain the list at any time when necessary.

Candidate Database Representation : candidate(list)

where list represents the list of conditions in the candidate solution

- Knowledge Base Database

The name of the knowledge base is asserted in this database.The reason for introducing this database is to keep the actual name of the knowledge base for further use in a possible save operation.

Knowledge Base Database Representation : knowledge_base(string)

where string is the file name of the knowledge base

- Addition Database

This database is asserted for keeping track of whether a new addition to knowledge base is made or not. If there is an addition, the necessity for a save operation will be displayed by the help of this database.

Addition Database Representation : addition(integer)

where integer is actually a binary number

0 for no addition , 1 for new addition

- Reject Database

During execution the user has the option of rejecting the proposed solution. The name of the subclass which is proposed but rejected by the user is asserted to this database. In a certain knowledge base there can be more than one alternative having the same class and subclass with different relations. If the user rejects one of these alternatives, logically the other one should also be rejected. It may happen very easily that the relation for the second alternative is also satisfied which will result in proposing the same subclass to the user more than once. This database inhibits the proposing of the same subclass to the user .

Reject Database Representation : reject(subclass)

where subclass represents the name of the proposed alternative

- True, False, Unknown Databases

In these databases the response of the user on different facts are asserted. Respond over a fact can be positive, negative, or neutral so the index of that fact is asserted to True, False or Unknown Databases respectively. These responses have to be kept somewhere during the consultation stage, in the analysis of alternatives to prevent the same fact to be asked more than once. If a fact is a one which is asked then its index should reside in one of these three databases.

True Database Representation : true(no)

False Database Representation : false(no)

Unknown Database Representation : unknown(no)

where no represents the index of a fact

As a summary the Context of this study contains seven different databases:

1- Candidate Database : candidate(list)

2- Knowledge Base Database : knowledge_base(string)

3- Addition Database : addition(integer)

4- Reject Database : reject(subclass)

5- True Database : true(no)

6- False Database : false(database)

7- Unknown Database : unknown(no)

3.4. INFERENCE MECHANISM

The inference mechanism or knowledge processor, in this study, is designed to have a goal-driven approach to problem solving. This approach assumes that the solution exists in the knowledge base initially which is the backward-chaining approach in Expert Systems. If this assumption turns out to be false, then the system can update the knowledge base by extracting new knowledge on the non-existing solution. Problem solving mechanism starts by the definition of the initial state-space or goal to be reached. This state-space is named as 'class'. The inference path starting from the initial state-space to the goal state, where the actual solution is found, will be explained by defining the predicates used in the inference mechanism one by one. Prolog implementation for the new program is given in Appendix A and the predicates forming the inference mechanism is explained in Appendix B.

3.5. EXPLANATION FACILITY

Explanation facility in this study, is designed such that the user has the option of getting information on the particular candidate solution that is being considered each time when faced with a question related to that solution. If the user desires to have explanation on how candidate solution and the question in consideration are related, this part of the program will supply the conditions satisfied and waiting to be satisfied for the candidate solution to be the actual solution. After providing this information the system returns to the question waiting for the response of the user.

3.6. KNOWLEDGE ACQUISITION

Knowledge acquisition operation can be defined as menu-driven program where the user has the option of acquiring knowledge on each of the relations in the knowledge base, namely Rule, Fact, Logical Relations, Logical Characteristics Databases. In knowledge acquisition operation information which is new is distinguished and added to the knowledge base.

IV. APPLIED KNOWLEDGE BASE

4.1. BACKGROUND OF THE KNOWLEDGE BASE

4.1.1. CONCEPT OF HISTORIC ARCHITECTURAL CONSERVATION

The concept of historic architectural conservation, which was taken as a museum related phenomenon in the beginning, changed significantly over the years. Nowadays it is taken as an adaptation for remunerative modern uses which is interpreted as a process of revitalization and integration of the properties having cultural and architectural values with economic and functional potentials [26].

The beginning and evolution of this concept and the approach taken towards conservation in Turkey are not as old and comprehensive as what one can observe in Europe [29]. Many valuable historical monuments and artifacts were lost during the Ottoman Imperial Period, because of the ignorance and apathy of the rulers and public in general [16]. At the beginning of the 19'th century, the voices of a few enlightened people, apparently influenced by the trends in Europe, started to come out. These voices, however, could not catch enough attention and so were ineffective [1]. After the foundation of the New Republic in 1923, Turkey entered a period of rapid change. The efforts to modernize and westernize the country on one hand, and the desire to erase the traces of the Ottoman culture on the other hand, influenced the approach taken towards conservation quite significantly [16]. During that period, this approach involved some efforts to determine and clarify the roots of

Turkish History and Anatolian Civilization apart from the Ottoman Peroid. The movable objects of value which are related to those civilizations were searched, found and taken into museums. Later on, this museum conservation was enlarged to encompass the concept of conservation of individual historic architectural monuments [1].

The concept of urban conservation in Turkey, however, is indeed quite recent. Around the 70's the authorities and the public were still solely interested in conservation of individual monumental buildings such as mosques, palaces and castles but not in group of houses or quarters. Over the past ten years there was a complete change of attitude towards what, in fact, needs to be protected. Planning and public authorities accepted the idea of historic and natural environment protection as much as the conservation of individual historic monuments and finally being influenced by the trends in Europe some related legislations, regulations and selecting criteria were accelerated. In some valuable historic areas, restricting building codes were determined and conservation plans were prepared'[7]. However all the efforts made, excluding a few examples, still could not reach the desired level. In most cases, a restricted line was drawn between the boundaries of a historic area declaring it as a prohibited zone.

By 1982, 417 Conservation areas, 100 of it being urban conservation areas, were designated. In these 417 conservation areas there were 3442 listed ancient monuments and 6815 listed historic buildings [11].

Today the integrated conservation approach, involving historical

archeological, architectural as well as social and economic aspects of saving revitalizing the urban areas worthy of conservation is the approach that is adopted and implied as permitted by the available resources [27].

4.1.2. EXISTING PROCEDURES AND RELATED LEGISLATIONS OVER CONSERVATION PLANNING DECISIONS

The conservation movement in Turkey is concieved as an integral part of the Urban Development Plan. According to the "Town Planning Act" (Law nr. 3194) Local authorities have to elaborate and implement urban development plans which predict development strategies, urban land uses and building regulations.

The current Law on "The Conservation of Cultural and Natural Entities Act" (Law nr. 2863 [17]) gives the Ministery of Culture and Tourism the responsibility for producing conservation decisions on historic and natural environment. This work is carried out by "The Supreme Council of Immobile Cultural and Natural Entities ". Local councils are established by the same law in order to carry out the designation of protection zones and cultural values which are to be approved and registered by the Supreme Council.

The first step taken in order to preserve a historic building or a site as an item worth of conservation, is surveying and documenting. The Supreme Council is responsible for the determination of cultural and natural values as well as designation of historic sites. After

surveying, documenting and approving , conservation items are recorded in the Local Land Registration Office. Listed buildings and designated sites are then taken into consideration in the preparation of the development plan of the settlement [28].

The Ministry of Culture and Tourism is in charge of producing conservation decisions on historic and natural environment, and Local Authorities ought to take into consideration those decisions when preparing and implementing Urban Development Plans [28].

In order to designate an object or a property as an item worth of conservation and ad-hoc . commitee of experts set up by the Ministry of Culture and Tourism has to identify and classify it as such, subject to the approval of Supreme Council of Cultural and Natural Entities [28].

4.1.3. PROBLEMS ASSOCIATED WITH IMPLEMENTATION OF CONSERVATION PLANNING DECISIONS

In a research carried out at Technical University of Istanbul among 51 Local Authorities various sorts of problems were identified in implementing the conservation decisions [29]. According to this research the main problems were the absence of effective conservation decisions, unqualified technical staff and contradiction between conservation decisions and planning strategies. Other problems were public reaction, lack of financial resources [29].

Many Local Authorities indicated that the conservation decisions turn out to be ineffective because of the long waiting time of planning

and implementation after the designation of an area as a conservation site. It is obvious that the delay in defining the limitations and obtaining the planning permissions would result in illegal demolition and illegal construction in most of the applications. Furthermore vague formulation of goals, objectives, and criteria for selecting and designating an area as a conservation site, and failure in combining conservation practices with economically viable activities would result in an increased scepticism of the people living those areas [29].

Shortage of specialized manpower turned out to be extreme in the same study. Out of 50 Local Authorities only 6 employed an urban planner and again out of that 50 only 25 employed an architect, the remaining 25 had only engineers or technicians. It was a real striking outcome that 25 Local Authorities did not have even an architect let alone a conservation expert [29]. So even though there is a tremendous amount of national cultural heritage in Turkey it is nearly impossible to meet the needs of conservation movement with this extreme shortage of specialized manpower.

The need for an Expert System to solve the problems related to ineffective conservation decisions and to replace specialized manpower turn out to be extremely important considering the results of the research.

4.2. DESIGN OF THE KNOWLEDGE BASE

The developed knowledge base aims to clarify the conservation type and degree depending on the characteristics of the buildings in consideration. As a result of this clarification the conservation rationale over buildings will be established so that these buildings will inherit their qualities and physical conditions over the years.

The decision criteria for conservation-type or conservation-degree are mainly the physical, structural, functional, and infra-structural conditions together with the value that the building has. The type of values are analyzed in three main headings: Cultural, Use, Emotional [5] (Figure 7). For any building to be a candidate of conservation it has to have certain values. Each value or combination of values and the existing conditions for other decision criteria have different effect on conservation decisions.

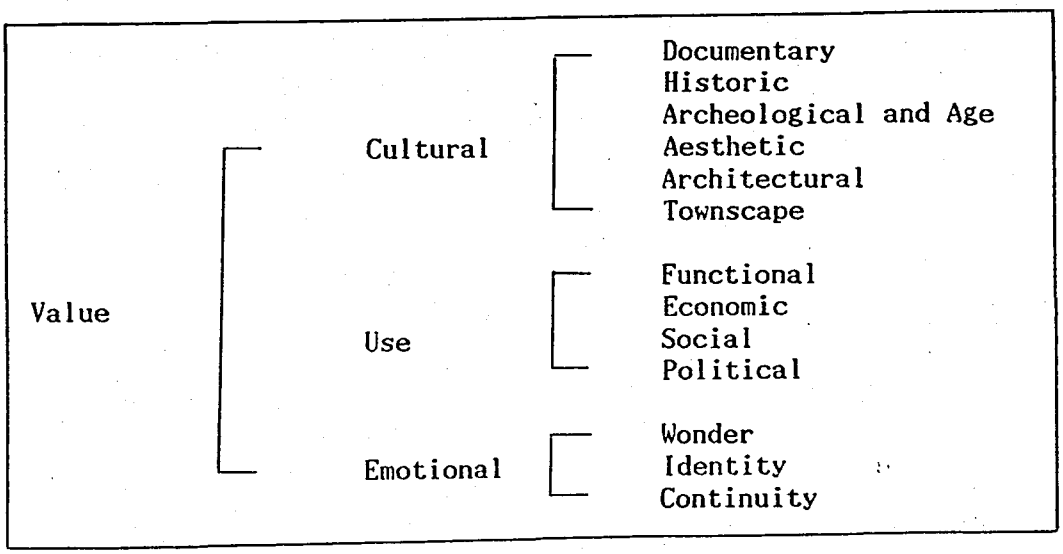

FIGURE 7 : Values Analyzed in Conservation Property

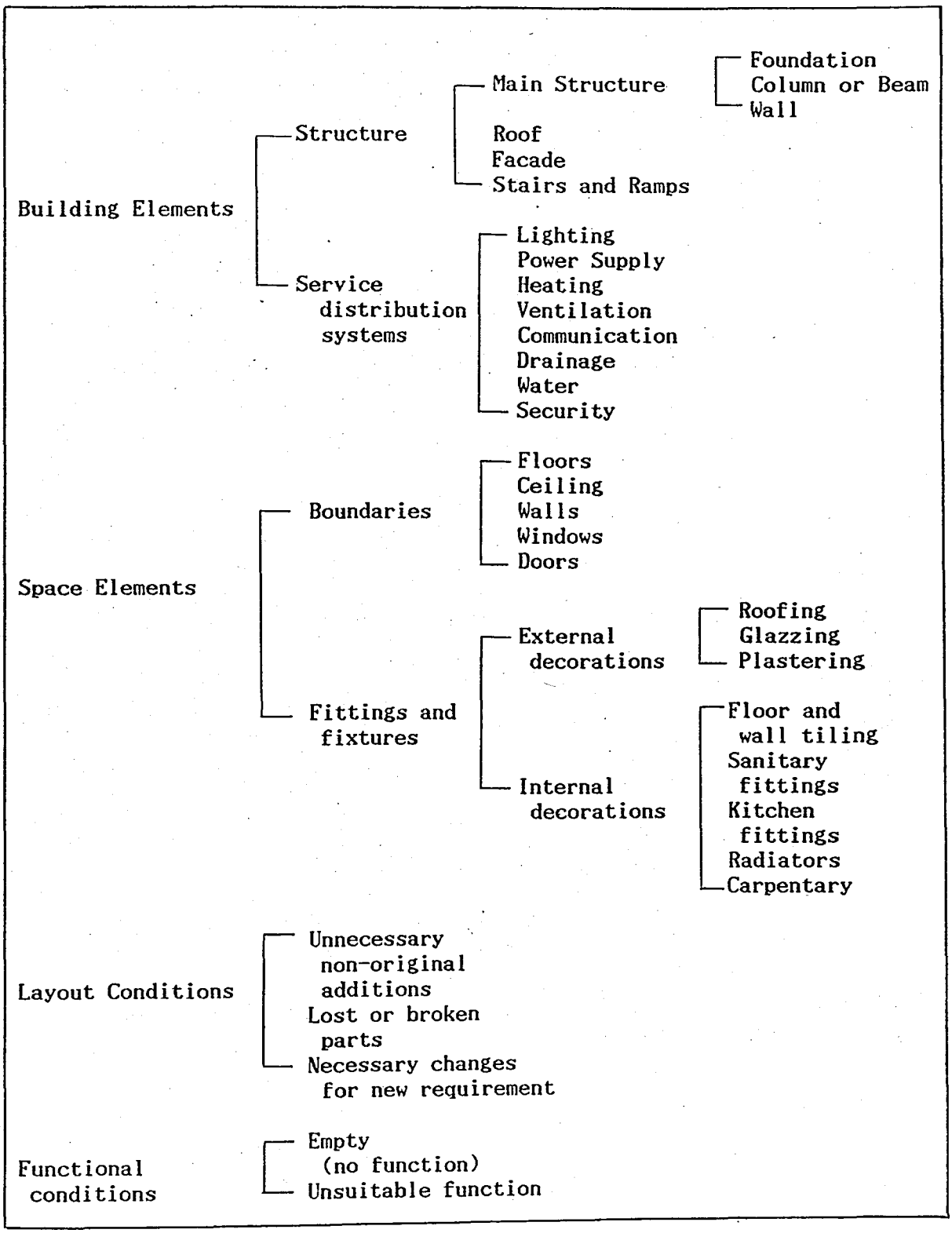

FIGURE 8 : Existing Conditions Analyzed in Conservation Property

Documentary, archeological and age, aesthetic, townscape, social, political, identity, and continuity values have high priority when compared to others namely the historic, architectural, functional, economic and wonder values. The buildings having at least one of the high priority values will surely be conserved. On the other hand buildings having low priority values may or may not be conserved depending on the existing conditions. Existing conditions are determined through the building, space, functional and layout conditions (Figure 8)

The existing conditions together with values result in different conservation-types, each embodying a different activity and conceptual emphasis. These include :

1- Prevention from Deterioration - Prevention from deterioration entails protecting cultural property by controlling its environment, thus preventing agents of decay and damage from becoming active. Prevention includes control of humidity, temperature and light, as well as measures to prevent fire, arson, theft and vandalism. In the industrial and urban environment it includes measures to reduce atmospheric pollution, traffic vibrations and ground subsidence due to many causes.

2- Preservation - Preservation deals directly with cultural property and aims to keep the entity in the same state. Damage and destruction caused by humidity, chemical agents, and all types of pests and micro-organisms must be stopped in order to preserve the object or sturucture. Maintenance, cleaning schedules, good housekeeping, and good management aid preservation.

3- Restoration - The object of restoration is to revive the original concept or legibility of the object. Restoration is frequently based upon respect for original material, archeological evidence, original design and authentic documents.

a- Reintegration - Reintegration is the replacement of broken or lost parts of the entity.

b- Consolidation - Consolidation is the physical addition or application of adhesive or supportive materials into the actual fabric of cultural property, in order to ensure its continued durability or structural integrity.

c- Liberation - Liberation is the removal of the un-necessary non-original additions from the cultural property.

4- Modernization - Modernization is to keep the historic buildings which are brought up to contemporary standards by providing modern amenities.

a- Revitalization - Revitalization is keeping the cultural entity in use by injecting new, suitable, functional and economic activities.

b- Conversion - Conversion is making physical and structural alterations in the cultural property for adapting to a new purpose.

c- Rehabilitation - Rehabilitation is repairing and renewing the existing infra-structure and hygienic conditions of a property to bring into a standard compatible with modern requirements of amenity and health.

5- Imitation - Imitation is making a copy of an original cultural property.

a- Reconstitution - Reconstitution is moving the entire building to new sites in order to prevent this valuable cultural property from being damaged irretrievably or threatened by its environment.

b- Reconstruction - Reconstruction is constructing the same cultural property again on the same site retaining its general characteristics.

c- Reproduction - Reproduction entails copying an extant artifact, often in order to replace some missing or decayed parts. A reproduction is thus often substituted in order to maintain the unity of a site or building.

Each conservation type has an associated conservation degrees that are grouped into three levels :

1- First Degree - First degree covers the buildings of exceptional interest.

2- Second Degree - Second degree covers the particularly important buildings.

3- Third Degree - Third degree covers the buildings of special interest, which warrant every effort being made to preserve them.

Fact, Logical Relations, Logical Characteristics, and Type Databases for the developed knowledge base is given in Appendix B.

V. MEASURE OF PERFORMANCE FOR THE DEVELOPED EXPERT SYSTEM

5.1. DOMAIN EXPERT'S EVALUATION OF THE SYSTEM

The domain expert was closely integrated with the system, during the development phase of the knowledge base; which means the state-space domain was directly extracted from her. She used the knowledge acquisition facility to acquire and update the knowledge base and found it useful.

During the revising phase the expert was satisfied with the inference mechanism and explanation facility. She used the explanation option quite frequently for checking decision making logic of the system. In a few cases she thought that the system did not respond correctly, but after selecting the explanation option she found out that, in fact, the system was right.

In the testing phase, she used 14 real-life cases in order to test the reliability of the system. In all cases she agreed with the results obtained. She named the system as suitable solution for the area of concern.

1-

2-

3-

4-

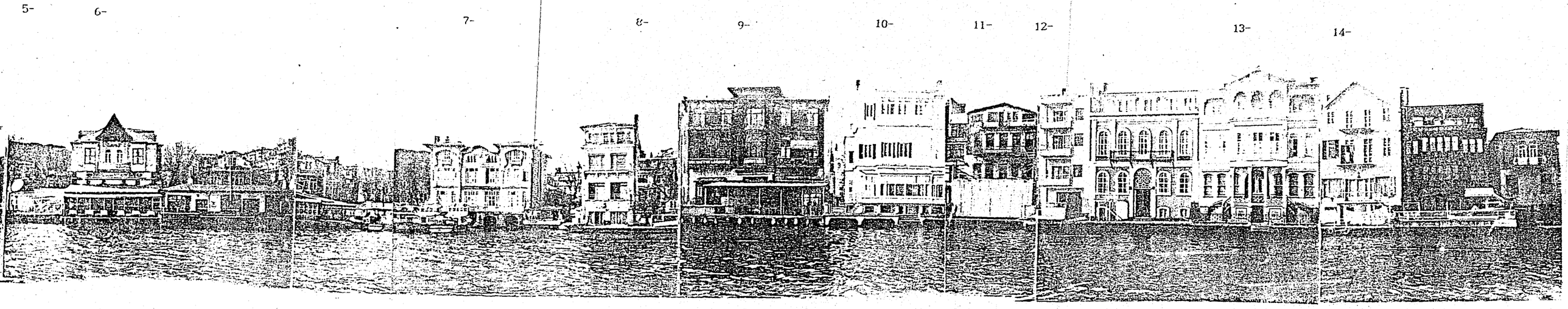

FIGURE 9. Silhouette of Tarabya - Yeniköy Coast

5.2. CASE STUDIES ON THE DEVELOPED KNOWLEDGE-BASE

Expert System developed is applied on 14 different properties along the Tarabya - Yenikoy Coast. The silhouette of these buildings are given in Figure 9. During the testing phase the characteristics of each building is entered to determine the related conservation-type and conservation-degree for each of the 14 cases. Domain expert later compared these solutions with her alternative solutions.

Case Study 1 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	NO
'has historic value'	NO
'has archeological or age value'	NO
'has aesthetic value'	NO
'has architectural value'	NO
'has townscape value'	NO
'has functional value'	NO
'has economic value'	NO
'has social value'	NO
'has political value'	NO
'has wonder value'	NO
'has identity value'	NO
'has continuity value'	NO

Conservation Type is NEW-BUILDING NEW-DESIGN

Case Study 2 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO
'has defective columns or beams'	YES
'has un-repairable column or beam defects'	NO
'has defective facade'	NO
'has defective stairs or ramps'	YES
'has un-repairable stair or ramp defects'	NO
'has defective roof'	YES
'has un-repairable roof defects'	NO
'has un-necessary non-original additions in the existing plan	NO
'has lost or broken parts from the original plan'	NO
'is necessary to introduce a new function'	YES
'is necessary to make changes in the original plan for new requirements'	YES
'has lacking or defective lighting facility'	NO
'has lacking or defective power supply facility'	YES

Conservation Type is CONSOLIDATION, REVITALIZATION, CONVERSION and REHABILITATION

Conservation Degree is SECOND DEGREE

Case Study 3 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	NO
'has historic value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO
'has defective columns or beams'	NO
'has defective facade'	NO
'has defective stairs or ramps'	NO
'has defective roof'	NO
'has un-necessary non-original additions in the existing plan	NO
'has lost or broken parts from the original plan'	NO
'is necessary to introduce a new function'	NO
'is necessary to make changes in the original plan for new requirements'	NO
'has lacking or defective lighting facility'	NO
'has lacking or defective power supply facility'	NO
'has lacking or defective heating facility'	NO
'has lacking or defective ventilation facility'	NO
'has lacking or defective drainage facility'	NO
'has lacking or defective water facility'	NO
'has lacking or defective communication facility'	NO

'has lacking or defective security facility' ..... NO

'has defective floors' ..... NO

'has defective ceiling' ..... NO

'has defective walls' ..... NO

'has defective windows' ..... NO

'has defective doors' ..... NO

'has defective roofing' ..... NO

'has defective glazzing' ..... NO

'has defective plastering' ..... NO

'has defective floor and wall tiling' ..... NO

'has defective sanitary fittings' ..... YES

Conservation Type is PRESERVATION

Conservation Degree is FIRST DEGREE

Case Study 4 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	NO
'has historic value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO
'has defective columns or beams'	NO
'has defective facade'	NO

'has defective stairs or ramps' ..... NO

'has defective roof' ..... NO

'has un-necessary non-original additions in the existing plan ..... NO

'has lost or broken parts from the original plan' ..... NO

'is necessary to introduce a new function' ..... YES

'is necessary to make changes in the original plan for new requirements' ..... YES

'has lacking or defective lighting facility' ..... NO

'has lacking or defective power supply facility' ..... NO

'has lacking or defective heating facility' ..... NO

'has lacking or defective ventilation facility' ..... NO

'has lacking or defective drainage facility' ..... NO

'has lacking or defective water facility' ..... NO

'has lacking or defective communication facility' ..... NO

'has lacking or defective security facility' ..... NO

Conservation Type is REVITALIZATION and CONVERSION

Conservation Degree is SECOND DEGREE

Case Study 5 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	NO
'has historic value'	NO
'has archeological or age value'	NO
'has aesthetic value'	NO

'has architectural value' ..... NO

'has townscape value' ..... NO

'has functional value' ..... YES

'is necessary or demanded to make a copy of the original building' ..... NO

'is faced with un-avoidable environmental conditions like erosion, flood, pollution' ..... NO

'has defective fountain' ..... NO

'has defective columns or beams' ..... YES

'has un-repairable column or beam defects' ..... YES

'has social value' ..... NO

'has political value' ..... NO

'has identity value' ..... NO

'has continuity value' ..... NO

Conservation Type is DEMOLITION

Conservation Degree is NO DEGREE

Case Study 6 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO

'has defective columns or beams' ..... YES

'has un-repairable column or beam defects' ..... NO

'has defective facade' ..... NO

'has defective stairs or ramps' ..... NO

'has defective roof' ..... YES

'has un-repairable roof defects' ..... NO

'has un-necessary non-original additions in the existing plan ..... NO

'has lost or broken parts from the original plan' ..... NO

'is necessary to introduce a new function' ..... NO

'is necessary to make changes in the original plan for new requirements' ..... YES

'has lacking or defective lighting facility' ..... YES

Conservation Type is CONSOLIDATION, CONVERSION and REHABILITATION

Conservation Degree is SECOND DEGREE

Case Study 7 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO
'has defective columns or beams'	NO

'has defective facade' ..... NO

'has defective stairs or ramps' ..... NO

'has defective roof' ..... NO

'has un-necessary non-original additions in the existing plan ..... NO

'has lost or broken parts from the original plan' ..... NO

'is necessary to introduce a new function' ..... NO

'is necessary to make changes in the original plan for new requirements' ..... NO

'has lacking or defective lighting facility' ..... NO

'has lacking or defective power supply facility' ..... NO

'has lacking or defective heating facility' ..... NO

'has lacking or defective ventilation facility' ..... NO

'has lacking or defective drainage facility' ..... NO

'has lacking or defective water facility' ..... YES

Conservation Type is REHABILITATION

Conservation Degree is FIRST DEGREE

Case Study 8 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	NO
'has historic value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like	

erosion, flood, pollution' ..... NO

'has defective fountain' ..... YES

'has un-repairable fountain defect' ..... YES

'has social value' ..... NO

'has political value' ..... NO

'has identity value' ..... NO

'has continuity value' ..... NO

Conservation Type is DEMOLITION

Conservation Degree is NO DEGREE

Case Study 9 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	NO
'has historic value'	NO
'has archeological or age value'	NO
'has aesthetic value'	NO
'has architectural value'	NO
'has townscape value'	NO
'has functional value'	NO
'has economic value'	NO
'has social value'	NO
'has political value'	NO
'has wonder value'	YES
'is necessary or demanded to make a copy of the original building'	NO

'is faced with un-avoidable environmental conditions like erosion, flood, pollution' ..... NO

'has defective fountain' ..... NO

'has defective columns or beams' ..... NO

'has defective facade' ..... NO

'has defective stairs or ramps' ..... YES

'has un-repairable stair or ramp defects' ..... NO

'has defective roof' ..... YES

'has un-repairable roof defects' ..... NO

'has un-necessary non-original additions in the existing plan ..... YES

'has lost or broken parts from the original plan' ..... YES

'is necessary to introduce a new function' ..... NO

'is necessary to make changes in the original plan for new requirements' ..... NO

'has lacking or defective lighting facility' ..... NO

'has lacking or defective power supply facility' ..... NO

'has lacking or defective heating facility' ..... NO

'has lacking or defective ventilation facility' ..... NO

'has lacking or defective drainage facility' ..... NO

'has lacking or defective water facility' ..... NO

'has lacking or defective communication facility' ..... NO

'has lacking or defective security facility' ..... NO

Conservation Type is CONSOLIDATION, LIBERATION and REINTEGRATION

Conservation Degree is SECOND DEGREE

Case Study 10 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	NO
'has historic value'	NO
'has archeological or age value'	NO
'has aesthetic value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO
'has defective columns or beams'	NO
'has defective facade'	NO
'has defective stairs or ramps'	NO
'has defective roof'	YES
'has un-repairable roof defects'	NO
'has un-necessary non-original additions in the existing plan	YES
'has lost or broken parts from the original plan'	NO
'is necessary to introduce a new function'	NO
'is necessary to make changes in the original plan for new requirements'	NO
'has lacking or defective lighting facility'	NO
'has lacking or defective power supply facility'	NO
'has lacking or defective heating facility'	NO
'has lacking or defective ventilation facility'	NO

'has lacking or defective drainage facility' ..... NO

'has lacking or defective water facility' ..... NO

'has lacking or defective communication facility' ..... NO

'has lacking or defective security facility' ..... YES

Conservation Type is CONSOLIDATION, LIBERATION and REHABILITATION

Conservation Degree is SECOND DEGREE

Case Study 11 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO
'has defective columns or beams'	YES
'has un-repairable column or beam defects'	NO
'has defective facade'	YES
'has un-repairable facade defect'	NO
'has defective stairs or ramps'	YES
'has un-repairable stair or ramp defects'	NO
'has defective roof'	YES
'has un-repairable roof defects'	NO
'has un-necessary non-original additions in the existing plan	NO

'has lost or broken parts from the original plan' ..... NO

'is necessary to introduce a new function' ..... NO

'is necessary to make changes in the original plan for new requirements' ..... NO

'has lacking or defective lighting facility' ..... NO

'has lacking or defective power supply facility' ..... NO

'has lacking or defective heating facility' ..... NO

'has lacking or defective ventilation facility' ..... YES

Conservation Type is CONSOLIDATION and REHABILITATION

Conservation Degree is SECOND DEGREE

Case Study 12 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO
'has defective columns or beams'	NO
'has defective facade'	YES
'has un-repairable facade defect'	NO
'has defective stairs or ramps'	NO
'has defective roof'	YES
'has un-repairable roof defects'	NO

'has un-necessary non-original additions in the existing plan ..... NO

'has lost or broken parts from the original plan' ..... NO

'is necessary to introduce a new function' ..... YES

'is necessary to make changes in the original plan for new requirements' ..... NO

'has lacking or defective lighting facility' ..... NO

'has lacking or defective power supply facility' ..... NO

'has lacking or defective heating facility' ..... NO

'has lacking or defective ventilation facility' ..... NO

'has lacking or defective drainage facility' ..... NO

'has lacking or defective water facility' ..... NO

'has lacking or defective communication facility' ..... NO

'has lacking or defective security facility' ..... NO

Conservation Type is CONSOLIDATION and REVITALIZATION

Conservation Degree is SECOND DEGREE

Case Study 13 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO

'has defective columns or beams' ..... NO

'has defective facade' ..... NO

'has defective stairs or ramps' ..... NO

'has defective roof' ..... NO

'has un-necessary non-original additions in the existing plan ..... NO

'has lost or broken parts from the original plan' ..... NO

'is necessary to introduce a new function' ..... NO

'is necessary to make changes in the original plan for new requirements' ..... NO

'has lacking or defective lighting facility' ..... NO

'has lacking or defective power supply facility' ..... NO

'has lacking or defective heating facility' ..... NO

'has lacking or defective ventilation facility' ..... NO

'has lacking or defective drainage facility' ..... NO

'has lacking or defective water facility' ..... NO

'has lacking or defective communication facility' ..... NO

'has lacking or defective security facility' ..... NO

'has defective floors' ..... NO

'has defective ceiling' ..... NO

'has defective walls' ..... NO

'has defective windows' ..... NO

'has defective doors' ..... NO

'has defective roofing' ..... NO

'has defective glazzing' ..... NO

'has defective plastering' ..... NO

'has defective floor and wall tiling' ..... NO

'has defective sanitary fittings' ..... NO

'has defective kitchen fittings' ..... NO

'has defective radiators' ..... NO

'has defective carpentary' ..... NO

Conservation Type is PREVENTION FROM DETERIORATION

Conservation Degree is FIRST DEGREE

Case Study 14 -

QUESTIONS	RESPONSES
Is it true that it	
'has documentary value'	NO
'has historic value'	YES
'is necessary or demanded to make a copy of the original building'	NO
'is faced with un-avoidable environmental conditions like erosion, flood, pollution'	NO
'has defective fountain'	NO
'has defective columns or beams'	NO
'has defective facade'	NO
'has defective stairs or ramps'	NO
'has defective roof'	NO
'has un-necessary non-original additions in the existing plan	YES
'has lost or broken parts from the original plan'	NO
'is necessary to introduce a new function'	NO
'is necessary to make changes in the original plan for	

new requirements' ..... NO

'has lacking or defective lighting facility' ..... NO

'has lacking or defective power supply facility' ..... NO

'has lacking or defective heating facility' ..... NO

'has lacking or defective ventilation facility' ..... YES

Conservation Type is LIBERATION and REHABILITATION

Conservation Degree is SECOND DEGREE

VI. CONCLUSION

Studies in Expert Systems area are considerably new in the world. Even though there are important developments achieved in the area, the user needs are not yet satisfied completely. With the available tools on hand, it is very hard to develop general Expert System shell to cover all of the state-space domains. The whole area of Expert Systems technology is clearly, if not in its 'infancy', at least in its 'adolescence'. The available technology is somewhat difficult to comprehend particularly for potential users who may be only vaguely familiar with the use of computers.

This study, which tries to clarify the conservation type and degree of buildings, carried out in this new Expert Systems area should be considered as a starting point. It does not claim to be a perfect solution, but rather it is just an application in developing both some kind of reliable program (shell) and suitable state-space domain.

This Expert System application is done on the particular state-space domain to meet the need for solving problems related to ineffective conservation decisions and for replacing specialized manpower to some extend. This application defined to be succesful by the domain expert after testing the system with 14 different real-life cases.

The program and the knowledge base developed are totally independent of each other. This means that ideally this program can be used with different knowledge bases. On the other hand, since the

program is designed in co-ordination with the characteristics of the conservation knowledge base, it is most probable that certain problems will occur when knowledge with totally different characteristics is applied.

As a result of this study, it is found out that the most suitable problems to be solved by Expert Systems are those which are self-contained, relatively routine, and for which there is expertise available from human experts. The expert must be not only willing to provide the knowledge, but also prepared to commit time during the development phase. These systems should be viewed as a way to disseminate and distribute scarce expertise in order to speed up given tasks - thus, to 'help' rather than 'substitute'.

Very much like in the development of conventional systems, if the planning, development and testing phases are strictly carried out as defined in section 2.4 and if the user and experts work together with the developers, so called the knowledge engineers, then there is a good chance that most of the problems stated in section 2.7 are overcome, giving way to a successful Expert System implementation.

APPENDIX A. Prolog Implementation for the New Program

```
code=2000
domains
  class, subclass = string
  phrase = string
  no = integer
  list = no*
  response = string
  strlist = response*

database

/*KNOWLEDGE BASE*/

  f(class,subclass,list)
  fact(no,phrase)
  implies(no,no)
  implied(no)
  type(response)

/*CONTEXT*/

  candidate(list)
  knowledge_base(string)
  reject(subclass)
  true(no)
  unknown(no)
  false(no)
  addition(integer)

include "menu.pro"

predicates

/*USER INTERFACE*/

  run
  init1
  init2
  way(response)
  clear1
  clear2
  accept(response)

 /*load knowledge*/
  load_knowledge
  control(response)
```

```
 /*consult knowledge*/
  consult_knowledge
  type_list1(strlist)
  type_list2(strlist)
  path(response)

 /*save knowledge*/
  save_knowledge
  addition_check
  backup(string,response)

 /*acquire knowledge*/
  acquire_knowledge

/*INFERENCE MECHANISM*/

 /*process candidate solution*/
  find(class)
  eliminated(list)
  test(no,list)
  member(no,list)

 /*search for each cond of the candidate solution*/
  search(class,subclass,list)
  asked_previously(no)
  get_implied(no,list,integer)
  absolute(no,no)
  getresponse(response)
  select(class,subclass,no,list,response)
  impliedIn(no)
  add(no)
  confusion(no)

 /*learn about the unknown solution from the expert*/
  askAbout(class)
  getSpec(subclass,list,list)
  getTrue(list)
  getFalse(list)
  appendlist(strlist,strlist,strlist)
  writeknw(no)
  str_list(list,strlist)
  menu_false(integer,strlist,response,list,list)

/*EXPLANATION FACILITY*/

  clearscreen
  getunknown(list,list)
  explain(class,subclass,list,list)
  writechoice(class,subclass)
  writeunknown(list,integer)
  writeknown(list,integer)
  confirmation(subclass)
  command(no)
  appendno(list,list,list)
```

```
/*KNOWLEDGE ACQUISITION*/

  acquisition
  acquire(response)

 /*general*/
  fact_phrase(response,no)
  getCond(no,phrase)
  getCNo(no,no)
  condedit(integer,phrase,phrase)
  condconcat(integer,integer,phrase,phrase)
  spaceinsert(integer,phrase,phrase)

 /*rule database acquisition*/
  rule_know
  getNew(response.list,list)

 /*fact database acquisition*/
  fact_know

 /*logical relation database acquisition*/
  relation_know

 /*logical characteristics database acquisition*/
  character_know

clauses

/*MAIN PROGRAM*/

  run:- init1, init2.

  init1:- makewindow(1,27,0,"",0,0,24,80),
    makewindow(2,27,0,"",0,0,24,60).

  clear1:- retract(true(_)), fail.
  clear1:- retract(false(_)), fail.
  clear1:- retract(unknown(_)), fail.
  clear1:- retract(reject(_)), fail.
  clear1:- retract(candidate(_)), fail.
  clear1:- retract(addition(_)), fail.
  clear1.

  clear2:- retract(f(_,_,_)), fail.
  clear2:- retract(fact(_,_)), fail.
  clear2:- retract(implies(_,_)), fail.
  clear2:- retract(implied(_)), fail.
  clear2:- retract(type(_)), fail.
  clear2:- retract(knowledge_base(_)), fail.
  clear2.

  accept(Choice):- cursor(Row,_), menu(Row,60,[yes,no],Choice).
```

```
/*USER INTERFACE*/

  init2:- shiftwindow(1), clearwindow,
    menu(10,30,["Load Knowledge","Consult Knowledge","Save Knowledge",
    "Acquire Knowledge","Quit"],Choice),
    way(Choice), clearwindow.

  way("Load Knowledge"):- load_knowledge.
  way("Consult Knowledge"):- consult_knowledge.
  way("Save Knowledge"):- save_knowledge.
  way("Acquire Knowledge"):- acquire_knowledge.
  way("Quit"):- addition_check.

 /*load knowledge*/

  load_knowledge:- knowledge_base(F), cursor(12,2),
    write("There is already an existing Knowledge Base ",F), nl, nl,
    write("  Do you want me to Erase it ?"), accept(R), R="no", !,
    init2.
  load_knowledge:- clearwindow, clear1, clear2, cursor(12,2),
    write("What is the Name of the Knowledge Base: "),
    readln(F), control(F), assert(knowledge_base(F)),
    assert(addition(0)), consult(F), !, init2.

  control(F):- existfile(F), !.
  control(F):- nl, nl,
    write("Error : The knowledge base ",F," is not found"),
    clearscreen, load_knowledge.

 /*consult knowledge*/

  consult_knowledge:- knowledge_base(_), shiftwindow(2),
    type_list1(T), type_list2(T), appendlist([reset|T],[return],TL),
    menu(14,60,TL,Choice), !, path(Choice).
  consult_knowledge:- cursor(12,2),
    write("There is NO Knowledge Base loaded.."), nl, nl,
    write("  So I cannot go on with Consultation"), clearscreen, !,
    init2.

  type_list1([C|I]):- type(C), retract(type(C)), type_list1(I).
  type_list1([]).

  type_list2([C|I]):- assert(type(C)), type_list2(I).
  type_list2([]).

  path("reset"):- clearwindow, clear1, !, consult_knowledge.
  path("return"):- init2.
  path(Response):- type(X), Response = X, find(X).
```

```
/*save knowledge*/

save_knowledge:- clearwindow, knowledge_base(F), existfile(F),
  cursor(12,2), write("Do you want to have a back up copy of Old ",F),
  accept(Response), backup(F,Response), fail.
save_knowledge:- knowledge_base(F), retract(knowledge_base(F)),
  clear1, save(F), !, init2.

backup(F,yes):- fronttoken(F,".",F2), concat(F1,F2,F),
  concat(F1,"dbk",FF), renamefile(F,FF), !.
backup(F,yes):- concat(F,".dbk",FF), renamefile(F,FF), !.
backup(_,no).

addition_check:- addition(1), !, cursor(12,2),
  write("Some additions are made to the Knowledge Base"), nl, nl,
  write(" Do you want to save the New Knowledge"),
  accept(Response), Response = yes, !, save_knowledge.
addition_check.

/*acquire knowledge*/

acquire_knowledge:- knowledge_base(_), cursor(12,2),
  write("Will you acquire Knowledge to Existing Knowledge Base ?"),
  accept(Response), clearwindow, Response = yes, acquisition,!, init2.
acquire_knowledge:- knowledge_base(_), addition_check, clear2, fail.
acquire_knowledge:- cursor(12,2),
  write("What is the Name of the Knowledge Base: "), readln(F),
  assert(knowledge_base(F)), existfile(F), consult(F), fail.
acquire_knowledge:- acquisition, save_knowledge, !.


/*INFERENCE MECHANISM*/

/*process candidate solutions*/

find(X):- f(X,Y,L), not(reject(Y)), not(eliminated(L)),
  assert(candidate(L)), shiftwindow(2), search(X,Y,L), nl.
find(X):- askAbout(X).

eliminated(L):- false(NO), member(NO,L), !.
eliminated(L):- true(NO), not(test(NO,L)), !.

test(NO,L):- member(NO,L), !.
test(NO,L):- implies(NO,X), member(X,L), !.

member(X,[X|_]):- !.
member(X,[_|L]):- member(X,L).
```

```
/* search for each condition of the candidate solution*/

  search(X,Y,[]):- candidate(L), getunknown(L,R), nl, !,
    writeunknown(R,1), writechoice(X,Y), confirmation(Y).
  search(X,Y,[NO|L]):- absolute(NO,N), asked_previosly(N), !,
    search(X,Y,L).
  search(_,_,[NO|_]):- absolute(NO,N), implied(N), !,
    retract(candidate(_)), fail.
  search(X,Y,[NO|L]):- absolute(NO,N), fact(N,F),
    write("Is it true that it ",F," ? "),
    getresponse(Choice), !, select(X,Y,NO,L,Choice).

  asked_previously(N):- true(N), !.
  asked_previously(N):- false(N), !.
  asked_previously(N):- unknown(N), !.

  getimplied(_,_,1).
ilgetimplied(I,[],0):- II = -I, add(II), retract(candidate(_,_,_)), !, fa
  getimplied(I,[H|T],0):- absolute(H,HH), not(asked_previously(HH)),
    fact(HH,F), write("Is it true that it ",F," ? "),getresponse(Choice),
    select(_,_,H,_,Choice), add(I), !, getimplied(I,T,1).
  getimplied(I,[_|T],0):- getimplied(I,T,0).

  absolute(NO,NO):- NO >= 0, !.
  absolute(NO,N):- N = -NO .

  getresponse(Choice):-  Row = 14, Col = 60,
    menu(Row,Col,[yes,no,unknown,why],Choice), write(Choice), nl.

  select(X,Y,NO,L,yes):- NO>0, assert(true(NO)), impliedIn(NO), !,
    search(X,Y,L).
  select(_,_,NO,_,yes):- NO<0, N=-NO, assert(true(N)), impliedIn(N),
    retract(candidate(_)), fail, !.
  select(_,_,NO,_,no):-  NO>0, assert(false(NO)), N=-NO,impliedIn(N),
    retract(candidate(_)), fail,!.
  select(X,Y,NO,L,no):-  NO<0, N=-NO, assert(false(N)), impliedIn(NO),
    !, search(X,Y,L).
  select(X,Y,NO,L,unknown):- absolute(NO,N), assert(unknown(N)), !,
    search(X,Y,L).
  select(X,Y,NO,L,why):- candidate(LL), appendno(L1,[NO|L2],LL),
    explain(X,Y,L1,[NO|L2]), !, search(X,Y,[NO|L]).

  impliedIn(A):- implies(A,B), add(B).
  impliedIn(_).

  add(A):- A<0, AA = -A, not(false(AA)), !, not(confusion(A)),
    assert(false(AA)), impliedIn(A).
  add(A):- A>0, not(true(A)), !, not(confusion(A)),assert(true(A)),
    impliedIn(A).
  add(_).

  confusion(A):- A<0, AA=-A, true(AA), command(AA), consult_knowledge, !.
  confusion(A):- false(A), command(A), consult_knowledge, !.
```

```
/*learn about the unknown solution from the expert*/

 askAbout(X):- nl, write("I do not know enough about this ",X), nl,
   write("Please tell me what it is: "), readln(Y), upper_lower(YY,Y),
   getSpec(YY,[],S), upper_lower(YY,YYY), assert(f(X,YYY,S)),
   retract(addition(_)), assert(addition(1)), clearwindow, !.

 getSpec(Y,K,S):- true(_), clearwindow,
   write("Following TRUE Facts are Known about ",Y,":"), nl,
   getTrue(T), appendno(K,T,SS), clearscreen, nl, !, getSpec(Y,SS,S).
 getSpec(Y,K,S):- false(_), clearwindow,
   write("Following FALSE Facts are Known about ",Y,":"), nl,
   cursor(Row,_), getFalse(H), nl,
   write("Please CHOOSE which of the FALSE Facts are to be Included"),
   nl, write("in the Definition of ",Y), str_list(H,CList),
   menu_false(Row,CList,"",[],L), !, appendno(K,L,SS),
   nl, !, getSpec(Y,SS,S).
 getSpec(Y,K,S):- nl, write("Will New Facts be Added about ",Y,"?"),
   cursor(Row,_), accept(Response), Response = yes, clearwindow,
   menu(Row,60,["true fact","false fact","return"],Choice),!,
   getNew(Choice,K,S), !.
 getSpec(_,K,K).

 str_list([NO|R],[C|L]):- str_int(Str,NO), concat("Fact ",Str,C),
   str_list(R,L).
 str_list([],[]).

 menu_false(_,_,"Return",IList,IList):- !.
 menu_false(Row,CList,"",IList,FList):-
   menu(Row,60,["Return"|CList],Choice),
   menu_false(Row,CList,Choice,IList,FList).
 menu_false(Row,CList,Choice,IList,FList):- concat("Cond ",Str,Choice),
   str_int(Str,NO), N = -NO, appendlist(C1,[Choice|C2],Clist),
   appendlist(C1,C2,C3), menu_false(Row,C3,"",[N|IList],FList).

 appendlist([],L,L).
 appendlist([X|L1],L2,[X|L3]):- appendlist(L1,L2,L3).

 writeknw(NO):-fact(NO,F), cursor(Row,_), write("   Fact ",NO),
   cursor(Row,12), write(": It ",F), nl.

 getTrue([NO|L]):- retract(true(NO)), writeknw(NO), getTrue(L).
 getTrue([]).

 getFalse([NO|L]):- retract(false(NO)), writeknw(NO), getFalse(L).
 getFalse([]).
```

```
/*EXPLANATION FACILITY*/

  getunknown([],[]):- !.
  getunknown([N|L],[N|Rest]):- N<0, NO=-N, unknown(NO),
    getunknown(L,Rest).
  getunknown([N|L],[N|Rest]):- unknown(N), getunknown(L,Rest).
  getunknown([_|L],R):- getunknown(L,R).

  clearscreen:-nl,nl, write("    PRESS any key to continue"),
    readchar(_), clearwindow.

  explain(X,Y,[],[NO|L2]):-clearwindow, nl,
    write("I AM TRYING TO SHOW THAT"), nl, writeunknown([NO|L2],1),
    writechoice(X,Y).
  explain(X,Y,L1,[NO|L2]):-clearwindow, nl, writeknown(L1,1),
    write("IS WHAT I KNOW ALREADY "), clearscreen,
    write("NOW, I AM TRYING TO SHOW THAT"), nl,
    writeunknown([NO|L2],1), writechoice(X,Y), clearscreen.

  writechoice(X,Y):- nl, upper_lower(O,X), upper_lower(P,Y),
    write(O," IS PROBABLY ",P), nl,nl.

  writeunknown([],_).
  writeunknown(R,20):- clearscreen, writeunknown(R,1).
  writeunknown([R1|R],I):- R1<0, R2=-R1, fact(R2,F), cursor(Row1,_),
    write("    If cond : it ",F," is FALSE "), cursor(Row2,_),
    J =I+Row2-Row1, nl, writeunknown(R,J).
  writeunknown([R1|R],I):- fact(R1,F), cursor(Row1,_),
    write("    If cond : it ",F," is TRUE "), cursor(Row2,_),
    J =I+Row2-Row1, nl, writeunknown(R,J).

  writeknown([],_).
  writeknown(R,20):- clearscreen, writeknown(R,1).
  writeknown([R1|R],I):- R1<0, R2=-R1, fact(R2,F), cursor(Row1,_),
    write("    Cond    : it ",F," is FALSE "), cursor(Row2,_),
    J =I+Row2-Row1, nl, writeknown(R,J).
  writeknown([R1|R],I):- fact(R1,F), cursor(Row1,_),
    write("    Cond    : it ",F," is TRUE "), cursor(Row2,_),
    J =I+Row2-Row1, nl, writeknown(R,J).

  confirmation(_):- write("Am I Correct ? "), accept(Response),
    write(Response), Response = yes, clearwindow, !.
  confirmation(Y):- assert(reject(Y)), nl, nl, fail.

  command(A):- clearwindow, nl,nl,nl,
    write("INFORMATION SUPPLIED BY YOU CONTRADICTS"),nl,nl,
    write("You said or implied TRUE and FALSE to"), fact(A,F),
    write(" Condition : it ",F),nl,nl,
    write("A Condition can not be true and false at the same time"),
    nl,nl,
    write("I am sorry to say that I have to begin all over again !"),
    clearscreen.
```

```
appendno([],L,L).
appendno([X|L1],L2,[X|L3]):- appendno(L1,L2,L3).


/*KNOWLEDGE ACQUISITION*/

acquisition:- clearwindow,
  menu(10,30,["Rule Knowledge","Fact Knowledge",
  "Logical Relation","Logical Characteristic","Return"],Choice),
  acquire(Choice).

acquire("Rule Knowledge"):- rule_know, !, acquisition.
acquire("Fact Knowledge"):- fact_know, !, acquisition.
acquire("Logical Relation"):- relation_know, !, acquisition.
acquire("Logical Characteristic"):- character_know, !, acquisition.
acquire("Return"):- !.

/*general*/

fact_phrase(true,NO):-  write(" Fact  :     it "), cursor(Row,_),
  readln(F), condedit(Row,F,FF), getCond(NO,FF).
fact_phrase(false,N):-  write(" Fact  : not it "), cursor(Row,_),
  readln(F), condedit(Row,F,FF), getCond(NO,FF), N = -NO.

condedit(_,F,F):- str_len(F,Length), Length <= 41, !.
condedit(Row,F,F):- scr_char(Row,59,Ch), Ch = ' ', !.
condedit(Row,F,F):- Row = Row + 1, scr_char(Row,1,Ch), Ch = ' ', !.
condedit(Row,F,FF):- condconcat(Row,59,F,FF).

condconcat(Row,Col,F,FF):- scr_char(Row,Col,Ch), Ch <> ' ',
  C = Col-1, !, condconcat(Row,C,F,FF).
condconcat(_,Col,F,FF):- C = Col - 16, frontstr(C,F,S1,S2),
  spaceinsert(Col,S1,S3), !, concat(S3,S2,FF).

spaceinsert(Col,S1,S3):- Col < 59, concat(S1," ",S2), C = Col + 1,
  spaceinsert(C,S2,S3).
spaceinsert(_,S1,S3):- S3 = S1.

getCond(NO,F):- fact(NO,F),!.
getCond(NO,F):- getcno(1,NO), assert( fact(NO,F) ).

getcno(N,N):- not(fact(N,_)),!.
getcno(N,N1):- NO=N+1, getcno(NO,N1).
```

```
/*rule database acquisition*/

 rule_know:- clearwindow, cursor(10,2), write("What is the"),
   cursor(12,2), write("Class Name : "), readln(X),
   cursor(14,2), write("Subclass Name : "), readln(Y),
   cursor(16,2), write("Give the Conditions for ",X," ",Y," Relation"),
   nl, menu(16,60,["true fact","false fact","return"],Choice),
   getNew(Choice,[],S), not(f(X,Y,S)), assert(f(X,Y,S)),
   not(type(X)), assert(type(X)), fail.
 rule_know:- clearwindow, cursor(12,2),
   write("Will you continue with Rule Acquisition ? "),
   accept(Response), Response = yes, !, rule_know.
 rule_know.

 getNew("return",K,K):- !.
 getNew("true fact",K,S):- fact_phrase(true,NO), not(confusion(NO)),
   appendno(K,[NO],NN),
   menu(12,60,["true fact","false fact","return"],Choice),
   !, getNew(Choice,NN,S).
 getNew("false fact",K,S):- fact_phrase(false,N), not(confusion(N)),
   appendno(K,[N],NN),
   menu(12,60,["true fact","false fact","return"],Choice),
   !, getNew(Choice,NN,S).

/*fact database acquisition*/

 fact_know:- cursor(12,0), fact_phrase(true,_), fail.
 fact_know:- clearwindow, cursor(12,2),
   write("Will you continue with Fact Acquisition ? "),
   accept(Response), Response = yes, !, fact_know.
 fact_know.

/*logical relations database acquisition*/

 relation_know:- cursor(12,2),write("Define the implying fact"), nl,
   menu(12,60,["true","false"],Choice1),
   fact_phrase(Choice1,NO1), write("  Define the implied fact"), nl,
   menu(12,60,["true","false"],Choice2),
   fact_phrase(Choice2,NO2), not(implies(NO1,NO2)),
   assert(implies(NO1,NO2)), fail.
 relation_know:- clearwindow, cursor(12,2),
   write("Will you continue with Relation Acquisition ? "),
   accept(Response), Response = yes, !, relation_know.
 relation_know.

/*logical characteristics database acquisition*/

 character_know:- cursor(12,0), fact_phrase(true,NO), not(implied(NO)),
   assert(implied(NO)), fail.
 character_know:- clearwindow, cursor(12,2),
   write("Will you continue with Character Acquisition ? "),
   accept(Response), Response = yes, !, character_know.
 character_know.
```

APPENDIX B. Detailed Explanations of Predicates Forming the Inference Mechanism of the Program Developed

Find(class) - Class represents the initial state-space of knowledge on which the research is being done.

When predicate 'Find' is executed the system either can find a particular solution so proposes this solution, or cannot find one so asks the help of the user. In reaching the solution, 'Find' predicate takes the first alternative with the specified class (initial state-space) in sequence from the knowledge base. This particular alternative contains - a subclass as a solution and a list of conditions as characteristics related to the subclass. After taking the alternative from the knowledge base, sequence of operations start.

Not(reject) The first operation is to check whether any other alternative having the same subclass as a solution is 'rejected' by the user. To continue with the operations, this subclass should not be in the Reject Database. If it turns out to be that it is in the database then the alternative is eliminated, and another alternative in sequence is taken from the knowledge base.

Not (eliminated) The second operation is to check whether the list of conditions is an already 'eliminated' list. For a list to be not('eliminated'), it should include all the True facts and it should not include any of the False facts. If these two conditions are not satisfied alternative is eliminated and

another alternative is taken just like in 'reject' check.

If the alternative passes from the two checks without being eliminated, this means it can actually be a candidate solution. So the related list of conditions of the alternative is asserted in the Candidate Database. Further operation is to search each fact related to the conditions in the list. If the condition and the information obtained does not match then candidate solution is retracted from the Candidate Database and another alternative is taken from the knowledge base. If all the conditions turn out to be matched with the information taken, the candidate solution is displayed as the proposed solution to the user. The user may or may not accept the solution. If the solution is accepted, 'Find' execution stops and the system returns back to its initial state. If he does not accept the solution, system eliminates that alternative and tries to find out other alternatives.

Assert (candidate)

Search

Select

Doing these operations one after the other, if the system arrives at the end of the knowledge base and yet can not come up with an accepted solution, it asks the help of the user to define this not-found subclass. When the definition operation is completed the system adds this new knowledge to the knowledge base and returns to the initial state.

Askabout

Eliminated(list) - list represents the list of conditions of a particular alternative

When predicate 'Eliminated' is executed the system tries to find whether the list contains any of the related indices of False facts or does not contain any of the related indices of True facts.

False Fact ===> Fact should not be the characteristic of the alternative

True Fact ===> Fact has to be the characteristic of the alternative

====> An alternative should contain ALL (True Facts)
NO (False Facts)

False Member

Checking operation starts with the False facts. The index of a False fact from the False Database is taken and checked against the list. If this number is a member of the list, then execution stops with the list being 'eliminated'. Otherwise checking operation continues over False Database unless the index of a particular False fact turns out to be the member of the list.

If it is found that none of the False facts is a member of the list then the checking operation continues on True facts. This time the condition number of a True fact from the True Database is taken and checked against the list. If the number is not a member of the list or is not implied by any other fact whose index is in the list , the execution stops with the list being 'eliminated'. Otherwise the checking operation continues over True Database unless the number of a particular True fact turns out not to be the

member of the list or not to be implied in the list.

When all True facts are processed with the list still staying as not('eliminated') execution of predicate 'Eliminated' stops.

Test(no,list) - No represents a condition, list represents the list of conditions of a particular alternative.

When predicate 'Test' is executed the system tries to find if the tested condition, or another condition which is implied by the tested condition is an element of the list.

Member Implies & Member

Let - A and B be two conditions where A implies B.
- L be a list of conditions

Suppose - B is a member of L

Then A is also a member of L.

Member(no,list) - No represents a condition, list represents the list of conditions of a particular alternative.

When predicate 'Member' is excuted, the system tries to check whether the condition is actually an element of the list or not.

Search(class,subclass,list) - Class represents the state-space of knowledge on which research is being done, subclass

represents the solution alternative for the class, list represents the list of remaining conditions associated with the alternative.

When predicate 'Search' is executed the system tries to make research over all the conditions of the candidate alternative by using the information extracted or will be extracted from the user. 'Search' is a recursive predicate in which execution stops either when the search over the list of conditions finishes or when the search fails at some point related to some mismatch of one condition and the information obtained .

Initially an alternative which has passed from 'reject' and 'eliminated' checks comes to 'Search' predicate containing the full list of condition. These conditions are actually numbers which can be positive or negative. Negative number indicates that 'False' response, positive number indicates that 'True' response is needed for the alternative to stay as a candidate. Asking the facts related to each condition one by one and trying to confirm the idea behind each one by the user is the main operation in 'Search' predicate.

asked_ previously

In doing the search, some of the facts may turn out to be already asked or implied. In such a case there is no need for searching the same fact one more time; the thing to be done is to take the related condition out of the list and continue searching remaining facts.

Non-searched facts can be of two main types. The first type of facts are the ones on which the user can respond directly. Whereas the second type of facts can only be implied indirectly by responses to other facts. If in a list of conditions a fact which should be implied resides, for the search operation to continue, the condition implying that fact has to be met first. In such a case the search operation on the candidate solution switches to search on the list of implying conditions .

Implied

Get_implied

Once it is understood that the fact is non-implied and not asked previously, the search operation continues to get the user response on that fact. According to the response obtained the 'Search' predicate tries to find whether the response and the condition matches. If it turns out that there is a mismatch the search operation fails causing the candidate alternative to be eliminated.

Getresponse

Select

This whole process continues until all the facts are searched and the solution is presented to the user.

Writechoice

Asked_previously(no) - no represents the condition

When predicate 'Asked_previously' is executed the system tries to find out whether the fact related to the condition is asked previously to the user.

The way to check whether a fact is asked (or implied) or

not, is simply to search through True, False or Unknown Databases. When a fact is asked, according to the response obtained, its condition number is asserted to one of the True, False or Unknown Databases. Assuming that the fact is already asked (or implied) and knowing that the alternative is not eliminated yet, the negative or positive condition number would reside in False or True Database respectively. If it is not in any of the two then it must surely be in Unknown Database, otherwise this fact can not be asked (or implied) previously.

False
True
Unknown

Get_implied(no,list,integer) - no represents the fact index which should be implied, list represents the list of conditions which imply the fact, integer is actually a binary number 0 representing the failure, 1 representing the success of the implication.

When predicate 'Get-implied' is executed the system tries to find whether the fact which should be implied can be really implied by searching on the list of conditions that imply it

During the execution of this predicate the operation is simply to ask the user about the facts in the list that are not asked previously until one of the facts which imply the fact in consideration turn out to be satisfied. If it turns out to be that none of the facts are satisfied then the execution stops with the fact being not-implied.

Getresponse
Not(asked_previously)
Select

Getresponse(response) - response represents any of the four different responds taken from the user namely; yes, no, unknown, why.

When the predicate 'Getresponse' is executed the system tries to get a response from the user on the fact being searched by using a menu-driven method.

There are four alternatives for a response which are yes, no, unknown and why. The user may know that the presented fact is correct or false one, then he will select 'yes' or 'no' option. If he does not have any idea over the presented fact, then he will select 'unknown' option. If he wants to have some kind of explanation on why he is asked that particular question, he will select 'why' option; (Later he has to decide whether the fact is true, false or unknown). After he decides on how to respond to the fact, 'Getresponse' execution stops.

Select(class,subclass,no,list,response) - class represents the state-space of the knowledge on which research is being done, subclass represents the candidate solution for the class, no represents the searched condition, list represents the list of remaining non-searched conditions associated with the alternative, response represents the respond of the user on the fact related to the searched condition.

When predicate 'Select' is executed the system tries to evaluate the meaning of the user response on the fact which is related to the searched condition.

User responses are divided into two in nature. First type includes only the 'why' response. This response is for getting explanation on the reason of asking a particular fact. Second type of responses are 'yes', 'no' and 'unknown' which are grouped together because each one is a reponse directly on a fact.

Explain

If the response is 'why' then the user will be supplied by the name of the candidate subclass, information gathered on that subclass, and information to be gathered necessary for that subclass to be the actual solution. After this explanation the user has to give a response over the fact.

If the response is 'unknown', this means the user cannot decide whether the fact is true or false. The 'unknown' response cannot supply sufficient information whether the condition is satisfied or not. With an optimistic approach the solution alternative may be claimed as not to be eliminated since the condition is not proved to be non-existent. So after asserting the fact index in the Unknown Database, the remaining condition list is sent back to the 'Search' predicate for further research.

Assert (unknown)

Search

If the response is 'yes' or 'no', a check will be done for determining whether the condition, necessary for a solution

to stay as a candidate, matches with the response of the user on the fact related to the searched condition. The way to check this match is to compare the condition with the response. For a succesful match the response has to be 'no' when the condition sign is minus, or 'yes' when the condition sign is plus. In these cases, the remaining list

Search Assert ImpliedIn

is sent back to 'Search' predicate after asserting the fact index to the relevant True or False Database and checking if the condition implies other conditions. Otherwise in case of a mismatch, that is, response is 'yes' when condition sign is minus or 'no' when sign is plus, the solution alternative is eliminated after asserting the fact and checking for implications.

Fail Assert ImpliedIn

ImpliedIn(no) - no represents the condition which may imply other conditions.

When the predicate 'ImpliedIn' is executed the system tries to find whether a condition implies any other conditions. If it comes out to be that there is at least one condition which is implied then its related fact index is asserted to the True or False Database according to the direction of implication (according to the sign of condition implied).

Implies

Add

Add(no) - no represents a condition which is implied by another condition whose related fact index is to be asserted to True or False Database if not asserted previously.

When the predicate 'Add' is executed the system tries to assert the index of the fact related to a condition to the proper True or False Database.

If the condition is negative this means the index should be asserted in False Database. But before doing the assertion, a check should be done to learn whether the index already resides in any of the True or False Databases. If False

Not(false) Database has the number, then there is no need to assert the same number again so execution will stop at that point. However if True Database has the number this will cause a mismatch , since a fact can not be true and false at the

Confusion same time. In such a case an error signal will be given and the consultation will start all over again.

If the index is positive, then just the opposite situations will occur, namely index should be asserted to True

Not(true) Database. After the check operation if True Database has the number there is no need to assert the number again and no need to continue operation, but if False Database has the

Confusion number which will result in a mismatch, whole consultation will start from the beginning.

Assert(true or false) After the checks are over, meaning that the condition number does not reside in any of True of False Databases, assertion can take place. Once assertion operation finishes, the

ImpliedIn newly asserted index should be sent back to 'ImpliedIn' for detecting further implications resulting from it.

Askabout(class) - class represents the state-space of the knowledge on which research is being done

When predicate 'Askabout' is executed the system tries to learn knowledge from the domain expert. This part of the program is only for domain expert usage.

readln
getspec
assert(f)

Learning operation starts by obtaining the name of the subclass. After getting the name the operation continues for getting specifications on the class subclass relation. The extracted knowledge is asserted in the knowledge base as a last step of this operation.

Getspec(subclass,list,list) - subclass represents the solution presented by the expert on the state-space, first list represents initial list of conditions, second list represents final list of conditions related to the subclass

When predicate 'Getspec' is executed the system tries to get conditions which define the class (state-space) subclass (solution) relation.

'Getspec' is an iterative predicate used for three different operations. There is a very important assumption lying under this predicate, that is :

All True Facts have to be included in defining the Relation (this is because the expert have accepted the existance of

those Facts)

gettrue In the first part True Facts are shown just for reminding purposes. Expert has no chance of interfering the addition of these Facts as the characteristics of the relation.

getfalse In the second part False Facts are shown. There is not any restriction over False Facts; they may or may not be included

menu_false in the definition. So it is expert's choice to include any False Fact in the definition.

In these two parts facts that are already considered by the system are considered once more in defining the relation. On the other hand there may be non-considered facts both present and not present in the knowledge base. In the third part

getnew definition of the fact by the expert is needed. If the fact turns out to be not existing in the knowledge base, it is included.

The execution of this predicate stops after completing the definition of the rule.

APPENDIX C. Fact, Logical Relations, Logical Characteristics and Type Databases

Fact Database

```
fact(1,"has an original plan")
fact(2,"will be conserved")
fact(3,"can be conserved")
fact(4,"has documentary value")
fact(5,"has historic value")
fact(6,"has archaeological or age value")
fact(7,"has aesthetic value")
fact(8,"has architectural value")
fact(9,"has townscape value")
fact(10,"has functional value")
fact(11,"has economic value")
fact(12,"has social value")
fact(13,"has political value")
fact(14,"has wonder value")
fact(15,"has identity value")
fact(16,"has continuity value")
fact(17,"has defective structure")
fact(18,"has defective foundation")
fact(19,"has defective columns and beams")
fact(20,"has defective facade")
fact(21,"has defective stairs or ramps")
fact(22,"has defective roof")
fact(23,"has a repairable stucture defect")
fact(24,"has un_repairable foundation defect")
fact(25,"has un_repairable column or beam defect")
fact(26,"has un-repairable facade defect")
fact(27,"has un_repairable stair or ramp defect")
fact(28,"has un_repairable roof defect")
fact(29,"is necessary to make a change in the infra_structure")
fact(30,"has lacking or defective lighting facility")
fact(31,"has lacking or defective power supply facility")
fact(32,"has lacking or defective heating facility")
fact(33,"has lacking or defective ventilation facility")
fact(34,"has lacking or defective drainage facility")
fact(35,"has lacking or defective water facility")
fact(36,"has lacking or defective communication facility")
fact(37,"is necessary to make a repair")
fact(38,"has defective floors")
fact(39,"has defective ceiling")
fact(40,"has defective walls")
fact(41,"has defective windows")
fact(42,"has defective doors")
fact(43,"has defective roofing")
fact(44,"has defective glazzing")
fact(45,"has defective plastering")
fact(46,"has defective floor and wall tiling")
fact(47,"has defective sanitary fittings")
fact(48,"has defective kitchen fittings")
```

```
fact(49,"has defective radiators")
fact(50,"has defective carpentary")
fact(51,"has un_necessary non_original additions in the existing plan")
fact(52,"has lost or broken parts from the original plan")
fact(53,"is necessary to make a change for new requirements")
fact(54,"is necessary to introduce a new function")
fact(55,"is faced with un_avoidable environmental conditions like
        erosion, flood, pollution")
fact(56,"is necessary or demanded to make a copy of the original
        building")
```

Logical Relations Database

```
implies(4,1)
implies(5,1)
implies(6,1)
implies(7,1)
implies(8,1)
implies(9,1)
implies(10,1)
implies(11,1)
implies(12,1)
implies(13,1)
implies(14,1)
implies(15,1)
implies(16,1)
implies(4,2)
implies(5,3)
implies(6,2)
implies(7,2)
implies(8,3)
implies(9,2)
implies(10,3)
implies(11,3)
implies(12,2)
implies(13,2)
implies(14,2)
implies(15,2)
implies(16,2)
implies(18,17)
implies(19,17)
implies(20,17)
implies(21,17)
implies(22,17)
implies(24,-23)
implies(25,-23)
implies(26,-23)
implies(27,-23)
implies(28,-23)
implies(30,29)
implies(31,29)
implies(32,29)
implies(33,29)
```

```
implies(34,29)
implies(35,29)
implies(36,29)
implies(37,7)
implies(38,9)
implies(39,9)
implies(40,9)
implies(41,9)
implies(42,9)
implies(43,9)
implies(44,9)
implies(45,9)
implies(46,9)
implies(47,9)
implies(48,9)
implies(49,9)
implies(50,9)
```

Logical Characteristics Database

```
implied(2)
implied(3)
implied(17)
implied(23)
implied(29)
```

Type Database

```
type("conservation type")
type("conservation degree")
```

BIBLIOGRAPHY

1- AKCURA, N. "Turkiye ve Eski Eserler", Mimarlik, No.8, 1972.

2- BLASER, A. "Knowledge-Based Systems - Status and Trends", IBM-IEC International Customer Executive Seminar, La Hulpe, February 1987.

3- ELDEIB, H. K. "An Integrated Approach to OR, AI and CAD in System Design", ORSA/TIMS Joint National Meeting, Boston, 1985.

4- ELLINGER, A. "The Role of Expert Systems in Financial Services Industries", IBM-IEC International Customer Executive Seminar, La Hulpe, February 1987.

5- FEILDEN, B. M. Introduction to Conservation of Cultural Property, Unesco, Rome, 1979.

6- HARMON, P., D. King. Expert Systems, John Wiley & Sons Inc., USA, 1985.

7- IPEKOGLU, B. "Preservation Policies in Turkey", Government/Non-Government Relationships in Urban Development International Seminar, Istanbul, June 1984.

8- KAVRAKOGLU, I. "Interactive Decision Support Sytems for Enhancing Expert Judgement", FBE-EM 86/06, Institute for Graduate Studies in Science and Engineering, Bogazici University, August 1986.

9- LANDAVER, W. "Expert Systems in Banking", IBM-IEC International Customer Executive Seminar, La Hulpe, February 1987.

10- LIMA, I. G. "Expert Systems - The Management Issues", IBM-IEC International Customer Executive Seminar, La Hulpe, February 1987.

11- MADRAN, E., N. OZGONUL. "Planli Donemde (1963-1981) Tarihsel Cevrenin Degerlendirilmesinde Kamunun Yaklasimi", Turkiye Birinci Sehircilik Kongresi, METU, 1982.

12- MAGEL, K. "Expert System Selection of Human-Computer Interfaces", North-Dakota State University, ORSA/TIMS Joint National Meeting, Miami Beach, October 1986.

13- MAHER, M. L. "Problem Solving Using Expert System Techniques", EDRC-12-02-86, Department of Civil Engineering, Carnegie-Mellon University, September 1986.

14- MAHER, M. L., P. LONGINOS. "Development of an Expert System Shell for Engineering Design", EDRC-12-05-86, Department of Civil Engineering, Carnegie-Mellon University, September 1986.

15- MAHER M.L., D. SRIRAM, S. J. FENVES, "Tools and Techniques for Knowledge-Based Expert Systems for Engineering Design", DRC-12-22-84, Department of Civil Engineering, Carnegie-Mellon University, December 1984.

16- MUMCU, A. "Eski Eserler Hukuku ve Turkiye", Ankara Universitesi Hukuk Fakultesi Dergisi, Vol. 24, No. 3-4, 1969.

17- Official Gazette. "The Preservation of Cultural and Natural Entities Act", No. 2863, July 23rd 1983.

18- Official Gazette. "Town Planning Act", No.3194, May 9th 1985.

19- REHAK, D. R., S. J. FENVES. "Expert Sytems in Civil Engineering, Construction and Construction Robotics", DRC-12-18-84, Robotics Institute, Carnegie-Mellon University, December 1984.

20- ROBOTHAM, P. "International Financial Markets - Need for New Technology", IBM-IEC International Customer Executive Seminar, La Hulpe, February 1987.

21- de SARAM, H. Programming in Micro-Prolog, Ellis Horwood Limited, West-Sussex, 1985.

22- SIMONS, G. L. Expert Systems and Micros, The National Computing Center Limited, Manchester, 1985.

23- SRIRAM, D., R. BANARES-ALCANTARA, V. VENKATASUBRAMANIAN, A. WESTERBERG, M. RYCHENER, "Knowledge-Based Expert Systems : An Emerging Technology for CAD in Chemical Engineering", DRC-06-76-84, Design Research Center, Carnegie Mellon University, December 1984.

24- THESEN, A., O. OSTBERG, L. LEI. "Knowledge Acquisition for Expert Systems : A Survey and Analysis", Technical Report, Department of Industrial Engineering, University of Wisconsin, February 1987.

25- Turbo Prolog Owner's Handbook, Borland International Inc., USA, 1986.

26- ZEREN, N. "Historic Architectural Site Conservation in Turkey", Institute of Planning Studies, University of Nottingham, 1985.

27- ZEREN, N. "Tarihsel Kentlerin Korunmasinda Cagdas Yaklasimlarin Degerlendirilmesi", Mimarinin Son 25 Yili Semineri, Faculty of Architecture, ITU, 1984.

28- ZEREN, N. "Tarihsel ve Kulturel Cevre Korunmasi", Tarihsel Cevre Korunmasi Seminer Dizisi, Faculty of Architecture, ITU, 1981.

29- ZEREN, N. "Kentsel Alanlarda Alinan Koruma Kararlarinin Uygulanabilirligi", Ph. D. Dissertation, ITU, 1981.