CHAPTER – 4

THEORETICAL DEVELOPMENTS IN EXPERT SYSTEMS

Knowledge is a theoretical or a practical understanding of a subject or a domain. It is also a sum of what is currently known. It is symbolized with power. Those who possess knowledge are termed as ‘experts’. The term ‘domain expert’ is used for a person who has deep knowledge of both facts and rules and has strong practical experience in a particular domain. In general a domain expert is a skilful person who can do challenging domain specific tasks with ease, applying their expertise with an optimum level of accuracy that non-experts may find difficult to accomplish.

The human mental process of problem solving, decision making is too complex to be represented in the form of algorithms but, most experts are capable of expressing their knowledge in the form of ‘rules’**[Negnevitsky,2002].**

4.1. EXPERT SYSTEMS

Expert systems are the first truly useful applications of AI. These are the systems that can be made to hold the knowledge of a specific domain or area which has been solicited from human experts. The system is also embodied with facts and rules which the human experts use to reason and come up with decisions. The system then puts relevant questions, elicits answers from the users, collects information and uses it to apply rules and comes up with conclusions in the same manner as a human expert would. **[Golabchi., 2008].** The system can also be given the capability to provide explanations for the reached conclusions in the same manner, as a human expert would present his justifications **[HSu and Su, 1991; Wick and Slagle, 1989].**

4.2 ADVANTAGES AND LIMITATIONS OF EXPERT SYSTEMS

Knowledge Availability: Once the system is up and running then that particular domain of knowledge will be available for use anywhere, any time. In other words, expertise is available whenever the system is turned on. Each rule is an independent piece of knowledge and the very style of representing knowledge in the form of rules, leads to self-documentation of knowledge for later recall.

Cost effectiveness: Once the preliminary expenses are paid then, except for maintenance costs which are relatively small, the expert systems are very cost effective.

Integration: There is high risk factor in most of the exploration sites. Hence, this gives a very good reason to have expert systems designed, integrated with other pre-existing systems and functionally put to use. Doing integrated exploration helps decrease the risk factor. Expert systems can be built in almost any area of exploration: field geology, mineralogy, petrography, seismic acquisition, processing, interpretation, etc. By using expert systems, integrated exploration can be performed and better results can be obtained.

Regular knowledge update: After an expert system has been developed, it can be updated more easily and economically compared to training a new expert.

Dealing with incomplete and uncertain knowledge: Most rule base expert systems are capable of representing the incomplete and uncertain knowledge. There are advanced techniques available like fuzzy-sets etc which are useful in such situations.

Expert systems as trainers: Expert systems can be used as trainers and educators for new engineers since they can provide reasoning for all decisions.

However, there are some disadvantages also for rule-based expert systems.

Opaque relations between rules: Although, the individual production rules are relatively simple, and self-documented, their logical interactions within a large set of rules may be opaque. Rule-based systems make it difficult to observe how individual rules serve the overall strategy.

Ineffective search strategy: The inference engine applies an exhaustive search through all the production rules during each cycle. Expert systems with large set of rules (over 100 rules) can be slow, and thus large expert systems may be unusable for real-time applications.

Inability to learn: In general, the expert systems do not have ability to learn from the experience. Unlike, a human expert, who knows when to 'break the rules', an expert system cannot automatically modify its knowledge-base, or adjust existing rules or add new rules. The knowledge-engineer is still responsible to revise and maintain an expert system.

Currently, research is on to explore the number of ways in which expert systems can be put to use in various activities ranging from initial steps of exploration, to well- completion to enhanced oil recovery **[Sasikumar et al., 1993].**

4.3. STRUCTURE OF EXPERT SYSTEMS

In early seventies, Newell and Simon from Carnegie Mellon University proposed a production model for expert systems which serves as a foundation for modern day expert systems. The production model is based on the idea that humans solve problems by applying their knowledge (expressed as production rules) to a given problem represented by problem-specific information. The production rules are stored in the long term memory and problem-specific information is stored in the short-term memory **[Newell, 1969].** The Figure 4.1 shows the production model.

Source: Negnevitsky, Pearson Education, 2002

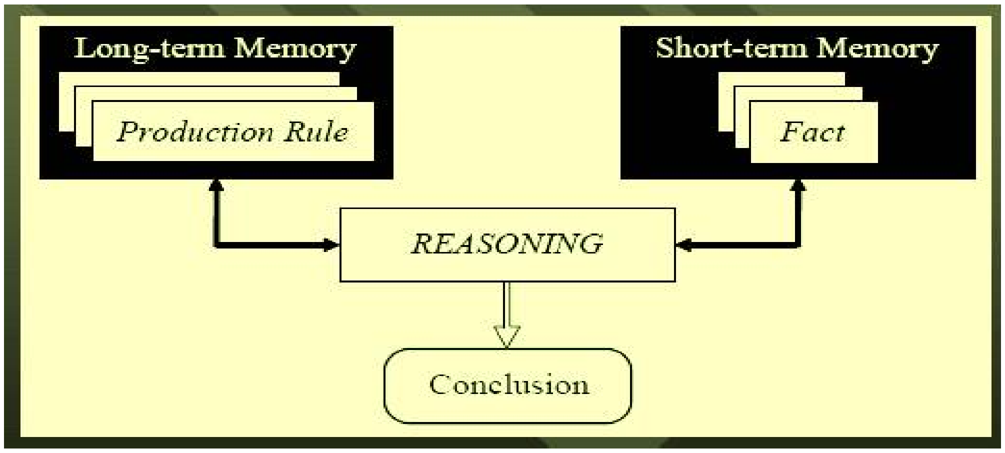


Figure 4.1 Production Model

Further the production model idea was enhanced into rule-based expert system. Figure 4.2 shows the structure of a typical rule-based expert system.

Source: Negnevitsky, Pearson Education, 2002

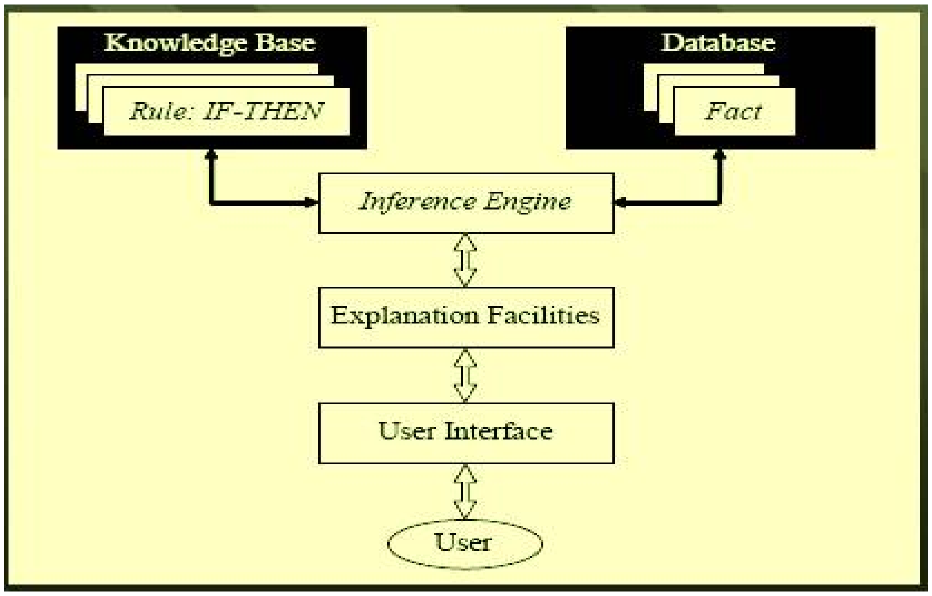


Figure 4.2 Structure of Rule-based Expert System

Expert systems consist of four main parts – the workspace, the knowledge base, the inference engine, and the explanation subsystem. Each plays an important role and these parts interface with each other in various complex ways.

4.3.1. Workspace/Working Memory

It is the space required for user interface, the area where knowledge extracted from knowledge base is temporarily kept, and the area where calculations are performed. Usually, the larger the workspace, the better the expert system operates.

4.3.2. Knowledge base

It is the part of the expert system that holds the knowledge of a particular domain. The knowledge representation can be done in different ways such as in the form of rule-base, in the form of frames, in form of semantic networks, in form of abstracts or in form of scripts.

4.3.2.1 'Rules' as a knowledge representation technique

The term 'rule' in artificial intelligence systems which is a very common knowledge representation technique can be defined as an *'if…….then'* structure that relates the given information or facts in the 'IF' part to some action in the 'THEN' part.

If (condition)
Then (action)

A rule provides some description of how to solve a problem. Rules are relatively easy to create and understand.

The '*if*' part is called an 'antecedent' or premise or the condition, and the '*then*' part is called the consequent or conclusion or an action. The antecedents are evaluated based on what is currently known about the problem being solved (contents of working memory/workspace). Each antecedent of a rule typically checks if the particular problem instance satisfies some condition.

A rule can have multiple antecedents, joined by the keywords, **AND** (conjunction), **OR** (disjunction) or a combination of both.

If *<Antecedent 1>* **AND** <*Antecedent 2*> **AND** *<Antecedent n>*

Then

<Consequent>

If *<Antecedent 1>* **OR** <*Antecedent 2*> **OR** *<Antecedent n>*

Then

<Consequent>

The antecedent of the rule incorporates two parts: an ***object (linguistic object)*** and its ***value***. The object and its value are linked by ***an operator***. The purpose of the operator is to identify the object and assign it the value. The operators such as '*is*', '*is not*', '*are*', '*are not*' are used to assign a symbolic value to the linguistic object. The expert systems can also use mathematical operators such as '>' (greater than), '<' (lesser than), to define an object as numerical and assign it to the numerical value.

The **consequents** of a rule typically alter the working memory (WM), to incorporate the information obtained by application of a rule. This could mean adding more elements to the WM, modifying an existing WM element or even deleting WM elements. They could also include actions such as reading input from a user, printing messages, accessing files, etc. When the consequents of a rule are executed, the rule is said to have been fired. Rules can represent relations, recommendations, directives, strategies and heuristics.

Sometimes the knowledge which is expressed in the form of rules is not known with certainty. In such cases, typically, a degree of uncertainty is attached to the rules. These degrees are called certainty factors **[Buchnan and Shortliffe, 1984].**

4.3.3. Inference Engine

It is the program part of an expert system. It represents a problem solving model which uses the rules in the knowledge base and the situation-specific knowledge in the WM to solve a problem. Given the contents of the WM, the inference engine determines the set of rules which should be considered. These are the rules for which the consequents match the current goal of the system. The set of rules which can be fired is called the conflict set. Out of the rules in the conflict set, the inference engine selects one rule based on some predefined criteria. This is called conflict resolution. A rule can be fired if all its antecedents are satisfied.

The matching of the antecedents to the facts produces inference chains (reasoning chains). The inference chain indicates how an expert system applies the rules to reach a conclusion. Figure 4.3 shows the match-fire procedure and Figure 4.4 exemplifies the reasoning chain.

Source: Negnevitsky, Pearson Education, 2002

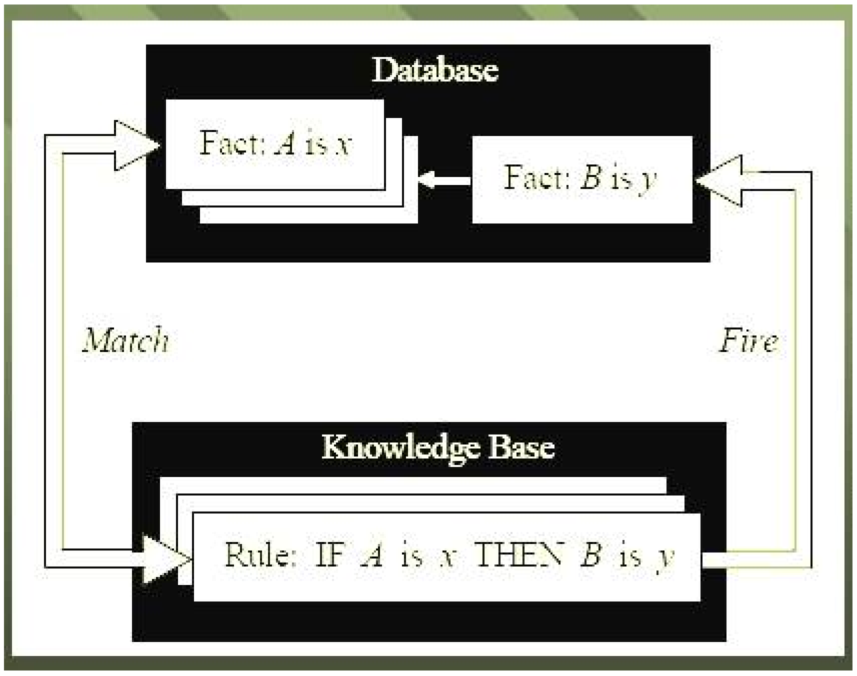
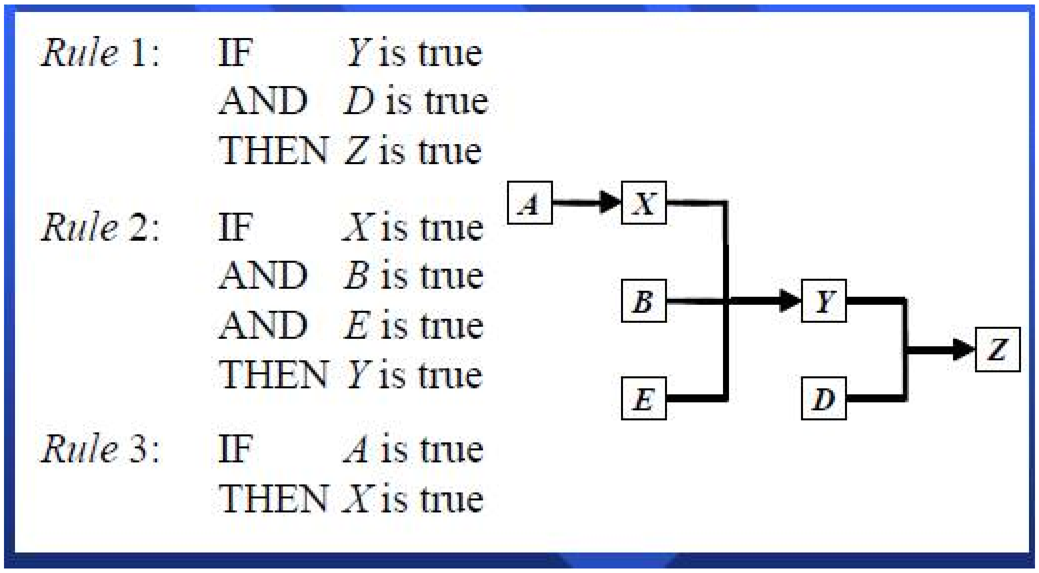


Figure 4.3 showing the match-fire procedure

Source: Negnevitsky, Pearson Education, 2002



Rule 1: IF Y is true
AND D is true
THEN Z is true

Rule 2: IF X is true
AND B is true
AND E is true
THEN Y is true

Rule 3: IF A is true
THEN X is true

Figure 4.4 Example of a reasoning chain

If the value of an antecedent is not known (in the WM), the system checks if there are any other rules with that as a consequent, thus setting up a sub-goal. If there are no rules for that antecedent, the user is prompted for the value and the value is added to the WM. If a new sub-goal has been set up, a new set of rules will be considered in the next cycle. This process is repeated till, in a given cycle, there are no sub-goals or alternatively, the goal of problem-solving has been derived. This inferencing strategy is called backward chaining, as it reasons backward from the goal to be derived. There is another strategy called forward chaining where the system works forward from the information it has in the working memory. In forward chaining, the conflict set will be created by rules which have their antecedents true in a given cycle. The system continues till the conflict set becomes empty.

Forward chaining methods are best for synthesis. If some of the elements of a particular case are known but not in its complete form, then this method can be used, profitably. If the end product is known, but the elements that make it up are not known, then backward chaining can be used for representation of knowledge. Backward chaining works in a goal oriented manner **[Krishnamoorthy and Rajeev, 1996].**

4.3.4. Explanation Subsystem

Expert systems typically need to be able to provide explanations regarding the conclusions they make. Most expert systems provide a mechanism whereby the user can ask questions about:

- Why a particular question is being asked?
- How the system came to a particular conclusion?

Providing explanations is essential in complex domains for user to understand how the system works and determine whether its reasoning is correct or not.

Typically the system keeps track of rules (knowledge), it is using and provides explanations based on a translation on these rules into English. Figure 4.5 shows the structure of a complete rule-based expert system.

Source: Negnevitsky, Pearson Education, 2002

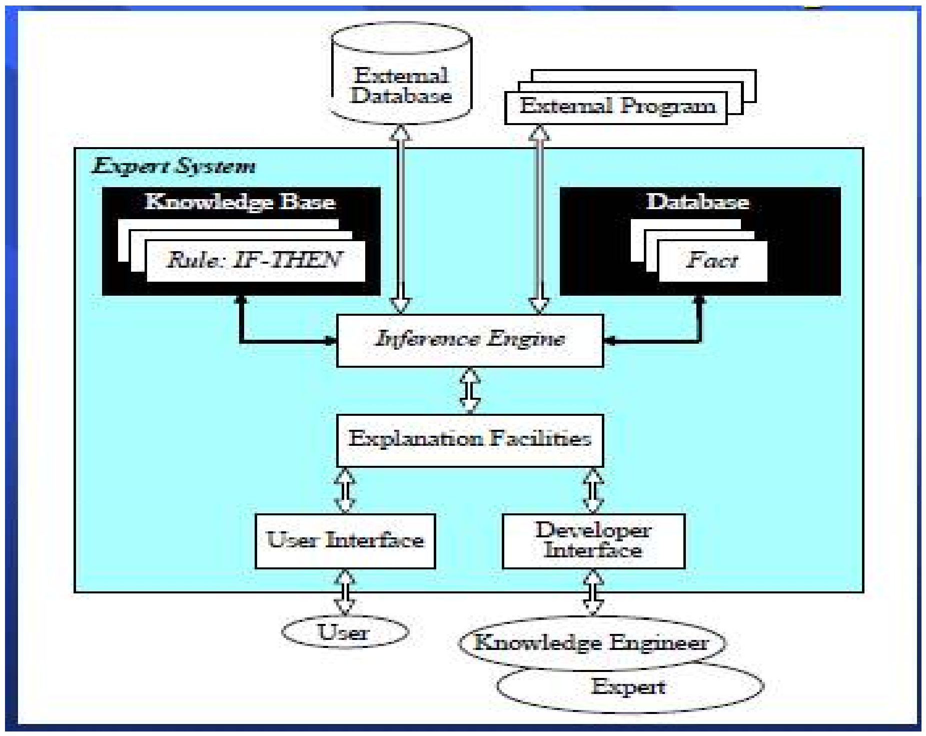
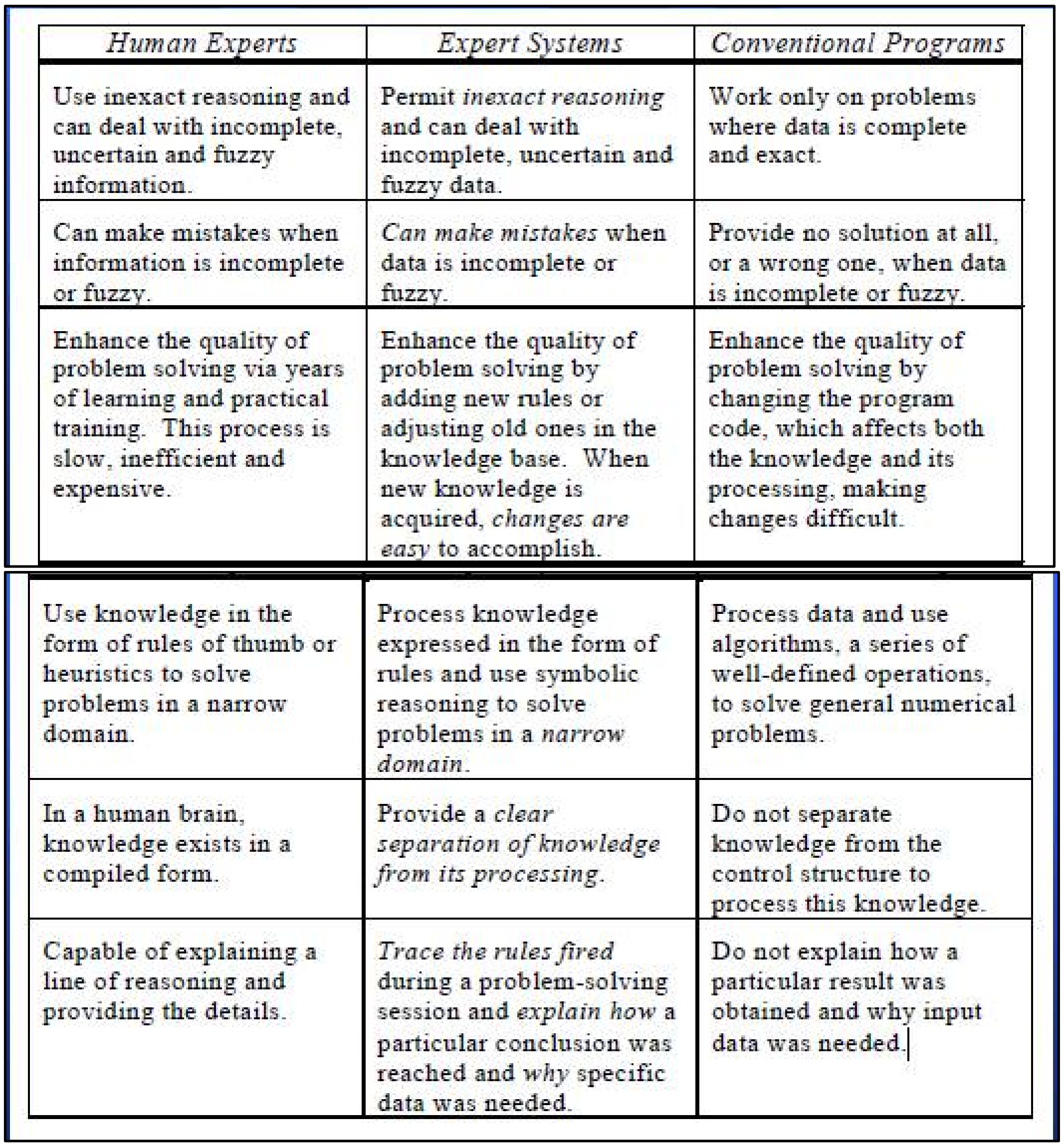


Figure 4.5 Structure of Expert System

In expert system, the knowledge is separated from its processing (the knowledge-base and the inference engine are split up). In case of other conventional computer programs, the knowledge and the control structures that process this knowledge are mixed up. This mixing up leads to difficulties in understanding and reviewing the program code, as any change to the program code, would affect both the knowledge and its processing **[Watermann, 1986].**

When expert system shells are used then the knowledge-engineer or an expert simply needs to enter rules into the knowledge-base. Each new rule adds new knowledge to the expert system and makes it smarter. Table 4.1 shows the comparison between human expert, an expert system and conventional computer programs.

Source: ***Negnevitsky, Pearson Education, 2002***



Human Experts	*Expert Systems*	*Conventional Programs*
Use inexact reasoning and can deal with incomplete, uncertain and fuzzy information.	Permit *inexact reasoning* and can deal with incomplete, uncertain and fuzzy data.	Work only on problems where data is complete and exact.
Can make mistakes when information is incomplete or fuzzy.	*Can make mistakes* when data is incomplete or fuzzy.	Provide no solution at all, or a wrong one, when data is incomplete or fuzzy.
Enhance the quality of problem solving via years of learning and practical training. This process is slow, inefficient and expensive.	Enhance the quality of problem solving by adding new rules or adjusting old ones in the knowledge base. When new knowledge is acquired, *changes are easy* to accomplish.	Enhance the quality of problem solving by changing the program code, which affects both the knowledge and its processing, making changes difficult.
Use knowledge in the form of rules of thumb or heuristics to solve problems in a narrow domain.	Process knowledge expressed in the form of rules and use symbolic reasoning to solve problems in a *narrow domain*.	Process data and use algorithms, a series of well-defined operations, to solve general numerical problems.
In a human brain, knowledge exists in a compiled form.	Provide a *clear separation of knowledge from its processing*.	Do not separate knowledge from the control structure to process this knowledge.
Capable of explaining a line of reasoning and providing the details.	*Trace the rules fired* during a problem-solving session and *explain how* a particular conclusion was reached and *why* specific data was needed.	Do not explain how a particular result was obtained and why input data was needed.

Table 4.1 Comparison between human expert, an expert system and a conventional computer program

4.4. BUILDING AN EXPERT SYSTEM

There are five members in the expert system development team. They are:

- 4.4.1. Domain Expert
- 4.4.2. Knowledge Engineer
- 4.4.3. Programmer
- 4.4.4. Project Manager
- 4.4.5. End user

The success of the developed expert system strongly depends on the extent of the team work done by them. Figure 4.6 shows the expert system development team **[Roth., 1983].**

***Source:* Negnevitsky, Pearson Education, 2002**

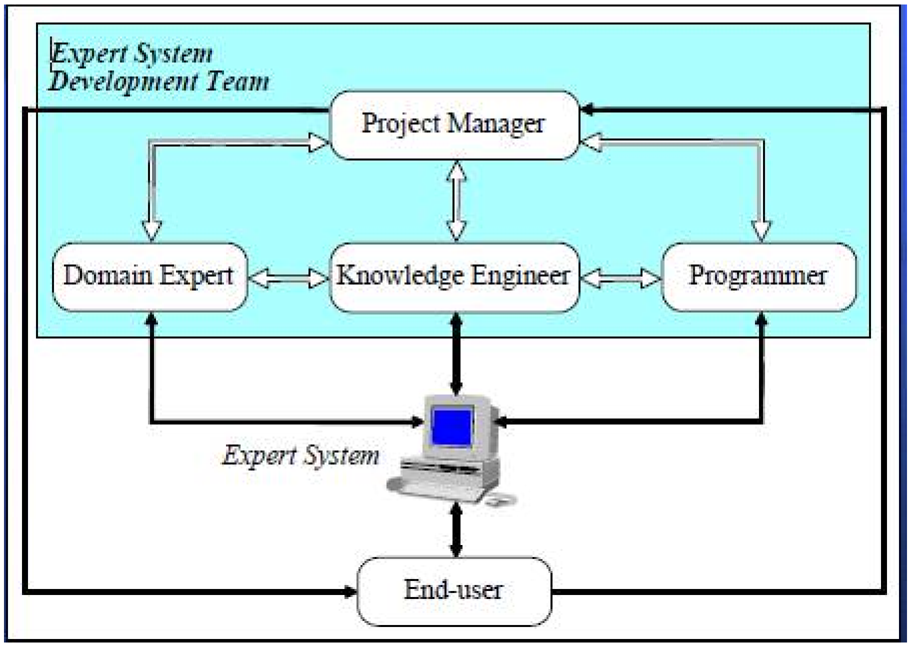


Figure 4.6 Team of Expert system development

4.4.1. Domain Expert

Domain expert being a knowledgeable and a skilful person, holding expertise in a particular domain is a very important asset in the whole process of expert system development. This expertise is to be captured in the expert system, hence the domain expert should be able to communicate his or her knowledge, be willing to participate in the expert system development and commit a substantial amount of time to the project.

4.4.2. Knowledge Engineer

Knowledge engineer is a person capable of designing, building and testing an expert system. He or she interviews the domain expert to find out, how a particular problem can be solved. The knowledge engineer establishes what reasoning methods the expert uses to handle facts and rules and decides how to represent them in the expert system. The knowledge engineer, then choose some software or an expert system development shell or looks at a programming language for encoding the domain knowledge. He is also finally responsible for testing, revising, integrating and deploying the expert system into the workplace.

4.4.3. Programmer

Programmer is the person responsible for the actual programming describing the domain knowledge in terms that a computer can understand. The programmer should have skills in symbolic programming languages such as LISP, Prolog, OPS5 and also some experience in application of expert system shells. In addition, the programmer should have proficiency in fundamental programming languages such as C, Pascal, BASIC and FORTRAN.

4.4.4 Project Manager

Project manager is the leader of expert system development team and responsible for keeping the project on track. He makes sure that all the deliverables and milestones are met, interacts with the knowledge engineer, domain expert, programmer and the end-user.

4.4.5. End user

End user is the person who is going to use the expert system, after it is developed. The user must not only be confident in expert system's performance but also feel comfortable using it. Therefore, the design of the user-interface of the expert system is also very vital for the project's success. The end-user's contribution in accepting the design is the crucial factor in the all over process.

4.5 PROGRAMMING LANGUAGES USED IN AI/EXPERT SYSTEMS

Expert systems can be written in any programming language but there are a certain set of languages which are termed as AI languages and these make the expert system designing and coding easy and fast. Some of the more common ones are LISP, PROLOG, and OPSS. Writing an expert system from scratch may require considerable amount of coding and hence it may become quite a time consuming endeavor. This can be made easy and development can be hastened by making use of specialized commercial packages called Expert System Shells.

4.6. EXPERT SYSTEM SHELLS

An expert system shell can be viewed as an expert system minus the domain knowledge. It allows knowledge of a domain to be encoded in a specific format and put into the system to create expert systems for different domains. The advantage of using a shell is that it avoids the need for extensive computer programming and allows the developer to focus only on the domain knowledge. This enables even non-computer professionals to create expert systems. **[Salim et al., 2003]**

There are a number of expert system shells free as well as commercially available in the market, which can be down loaded or bought off-the-shelf and customized by the users, as per their requirements. Few examples are: Aion, Acquire, Attar, Drools, Clips, Infosapient, Corticon, EXSYS, Jess, JLisa, ILOG Rules, Jena2, JEOPS, Mandarex, Mindbox, OFBiz, OpenRules, PegaRules, Flex and Pellet etc ***[http://www.kbsc.com/rulebase.html].***

.