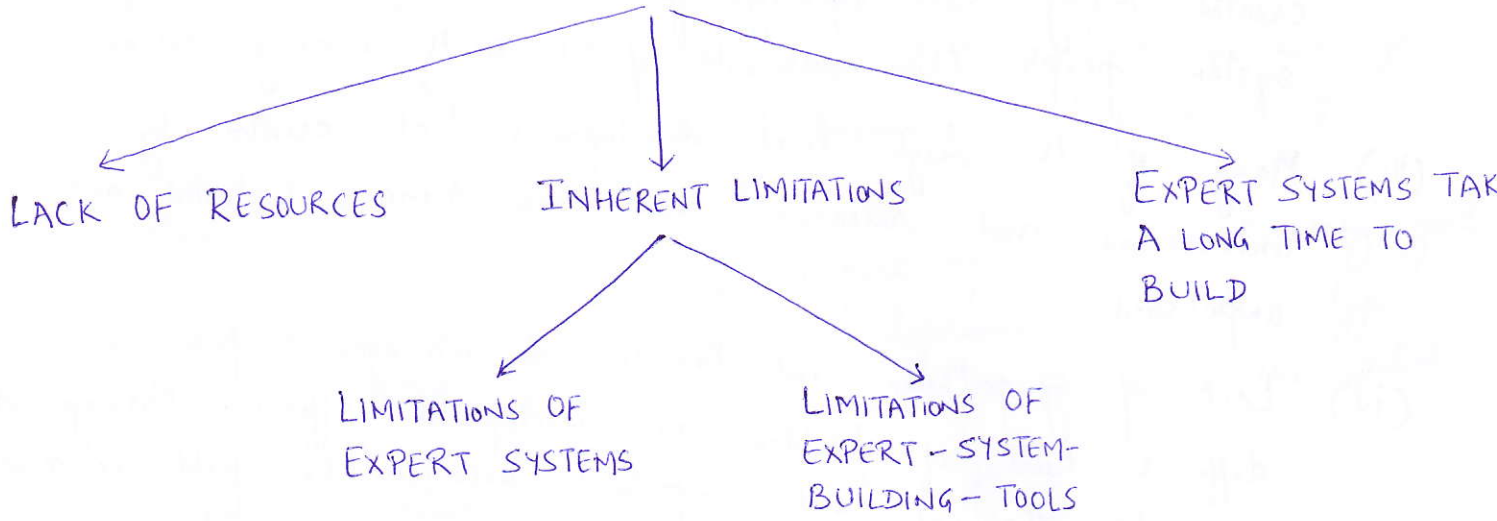
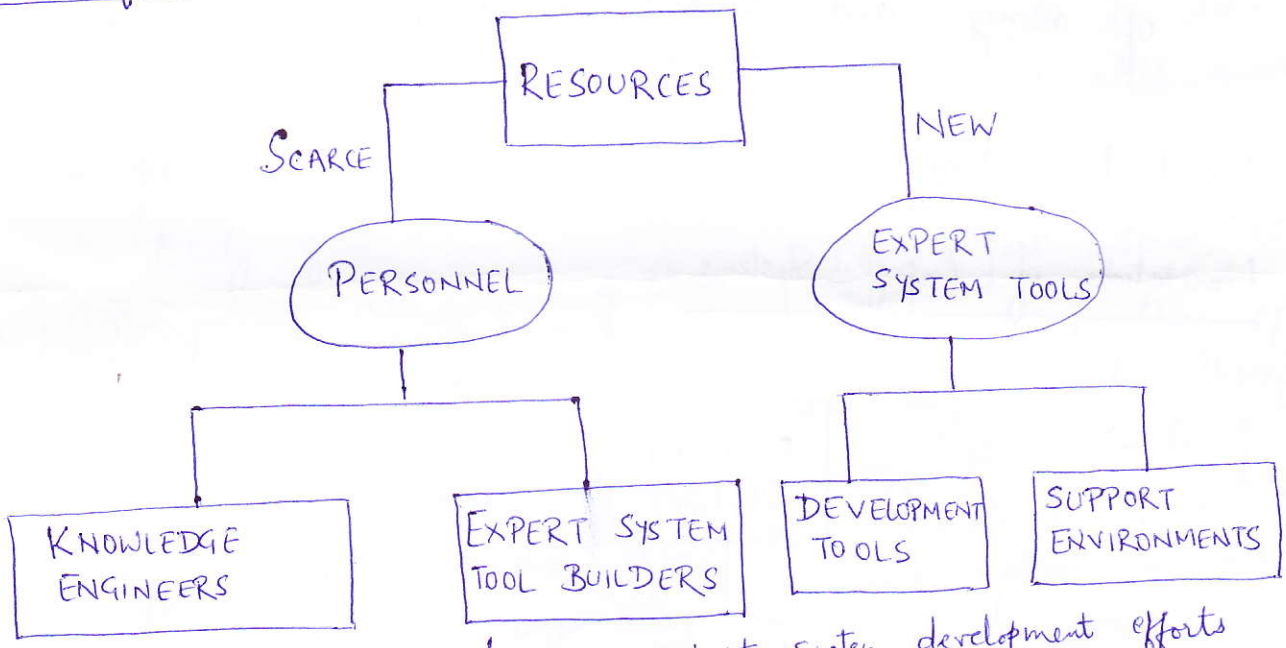
# UNIT - IV

## DIFFICULTIES IN DEVELOPING AN EXPERT SYSTEM :




### (1) Lack of Resources :




Lack of Resources hampers expert system development efforts

(a) Personnel : Personnel competent to design and develop the systems are scarce.

(b) Expert system tools : Few of the high-level support tools and languages are fully developed or reliable. Many of them are new and untested.

Reasons for Lack of Resources :

(i) Expert systems is new and unfamiliar to most computer specialists and somewhat difficult for them to understand and apply.

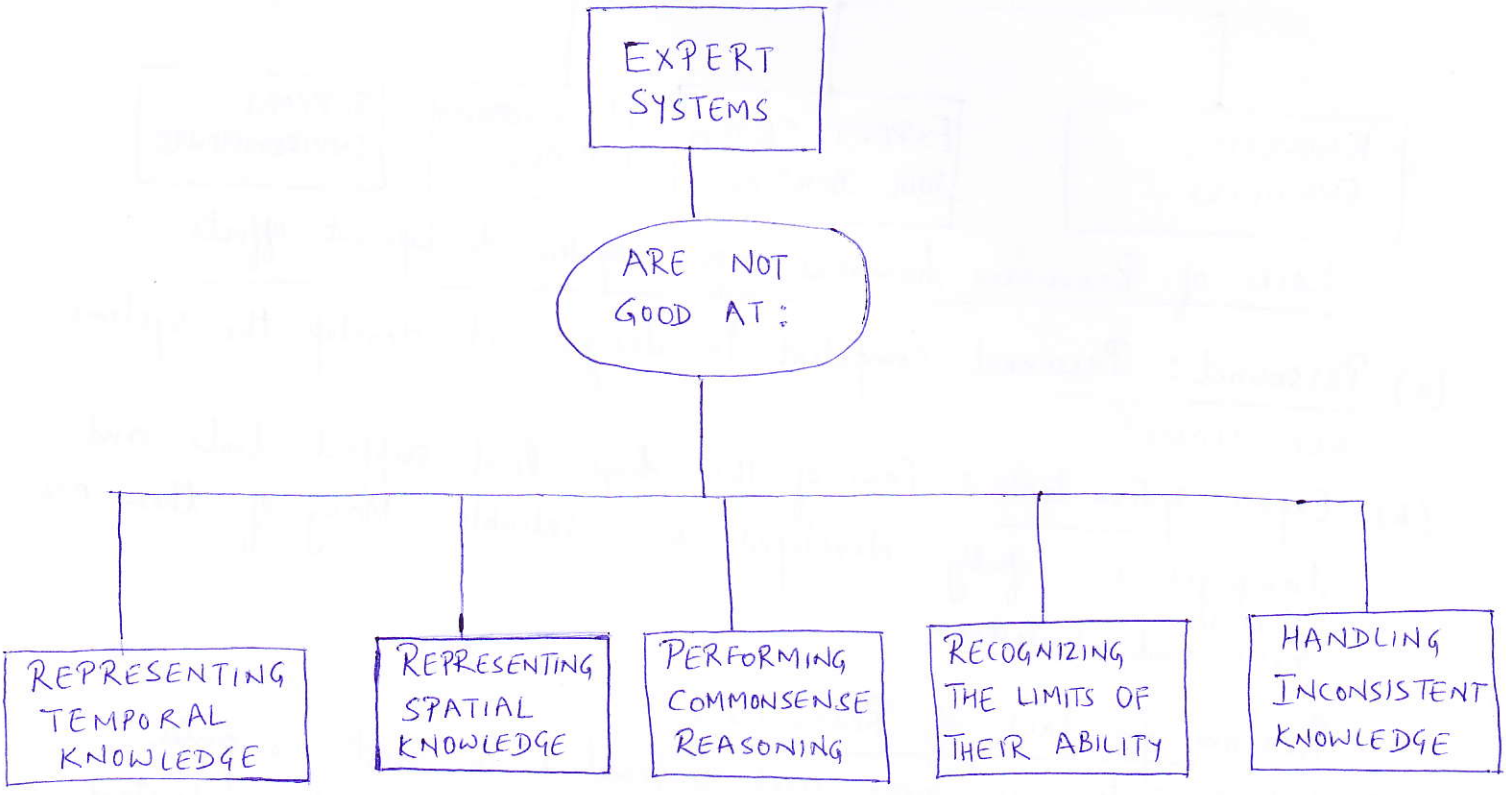
(ii) The crush of companies entering the AI arena has created many more openings for experienced AI and expert system people than can be filled with existing personnel.

(iii) Many of the high-level development tools created by universities and research institutions remain untested and unpolished.

(iv) Lack of sympathetic and knowledgeable management poses a different kind of problem in development efforts. Management can be skeptical and impatient because the field is new and building an expert system requires very large amou of money and time.

## (2) Inherent Limitations :

### (a) Limitations of Expert Systems :




Limitations of expert Systems

The limitations of expert systems are:

(i) Representing temporal Knowledge: Expert systems are not very good at representing temporal Knowledge. Representations of this type can require huge amounts of memory to keep track of the state of things at various points in time.

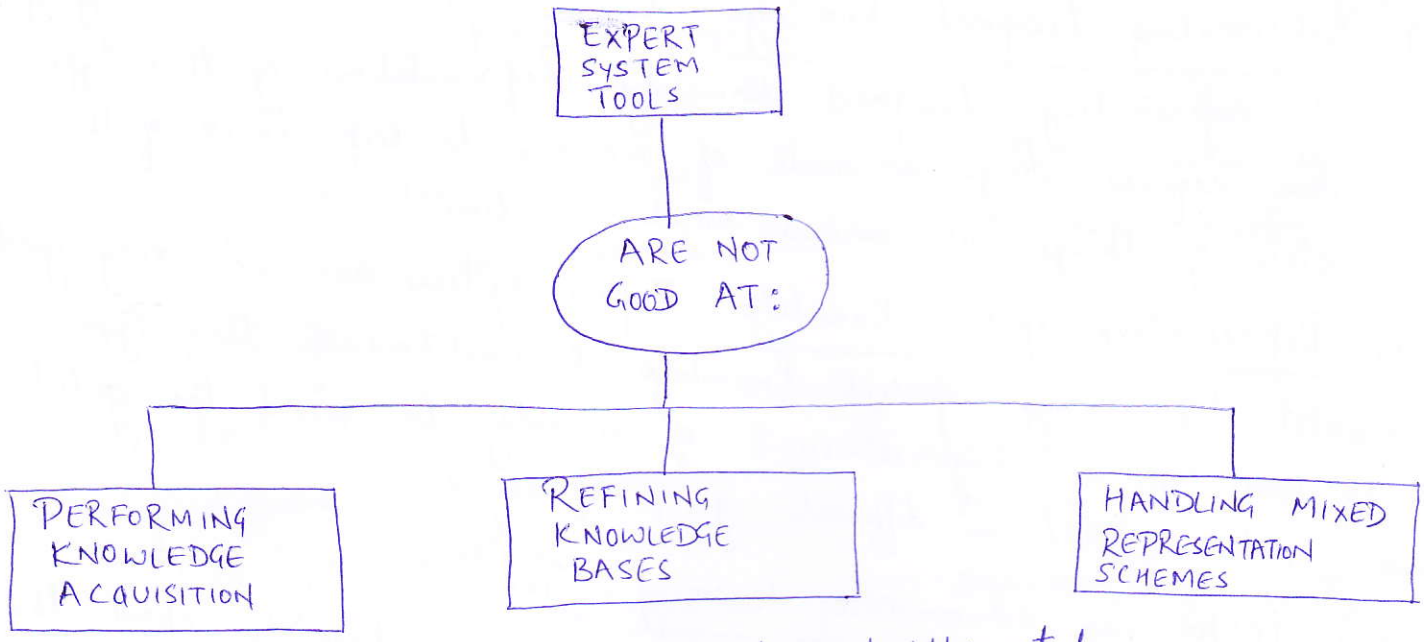
(ii) Representing spatial Knowledge: Expert systems are not very good at representing spatial Knowledge. Representations of this type can require huge amounts of memory to record the spatial relations between different groups of objects.

(iii) Performing commonsense reasoning: The expert systems have problems using commonsense or general Knowledge about the world. If this type of Knowledge is crucial to solving a problem, the Knowledge engineering approach will most likely fail.

(iv) Recognizing the limits of their ability: Expert systems have a very narrow domain of expertise and hence their operation is not as robust as the users might want. Therefore expert systems have difficulty recognizing the limits of their ability. When pushed beyond their limits or given problems for which they were designed, expert systems can fail.

(v) Handling inconsistent Knowledge: Expert systems have difficulty dealing with erroneous or inconsistent Knowledge because they rely on a body of rules that represent abstracted Knowledge of the domain and are not able to reason from basic principles to recognize incorrect Knowledge or reason about inconsistencies.

## (b) Limitations of Expert-System-Building-Tools:




### Limitations of expert-system-building tools

The limitations of expert-system-building tools are:

(i) Performing Knowledge acquisition: It is tedious and time-consuming to extract knowledge from an expert and incorporate it into a large knowledge base.

(ii) Refining Knowledge bases: The expert-system-building tools are inadequate to help refine and correct the expert system's knowledge base. It may take a large effort to get a small improvement in performance.

(iii) Handling mixed representation schemes: Languages for building expert systems are not as flexible and general as the knowledge engineer needs. Particular types of knowledge (e.g. temporal or spatial) cannot be represented easily or different representational schemes (e.g. rules and frames) cannot be represented naturally and efficiently in the same language. Many languages do not provide mechanisms for building adequate user-interfaces. The knowledge engineer needs sophisticated graphics and reliable natural language capabilities to make the user-system interaction smooth and efficient.

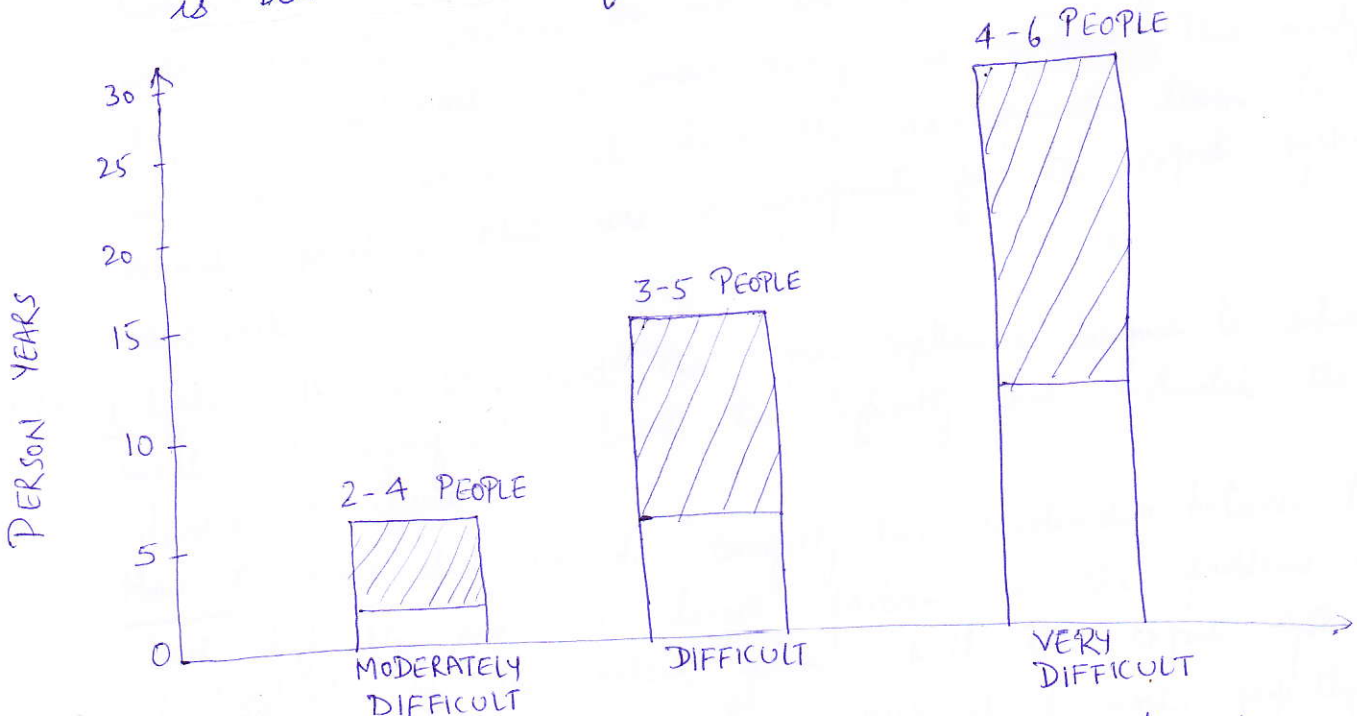
## (3) Expert Systems take a long time to build:

Expert systems cannot be built quickly. The time to build expert systems depends upon the type of problem. The development effort and time required to build an expert system varies according to the type of problem. There are basically three types of problems:

(a) Moderately difficult: It takes from five to ten person-years to build an expert system to solve a moderately complex problem. e.g. PUFF required about five person-years of effort. The time required to build a moderately difficult problem is between six months and one and one-half years at a two- to four-person level.

(b) Difficult: It takes about eight person-years to build an expert system to solve difficult problem. e.g. XCON. The time required to build a difficult problem is between one and one-half and three years at a three- to five-person level.

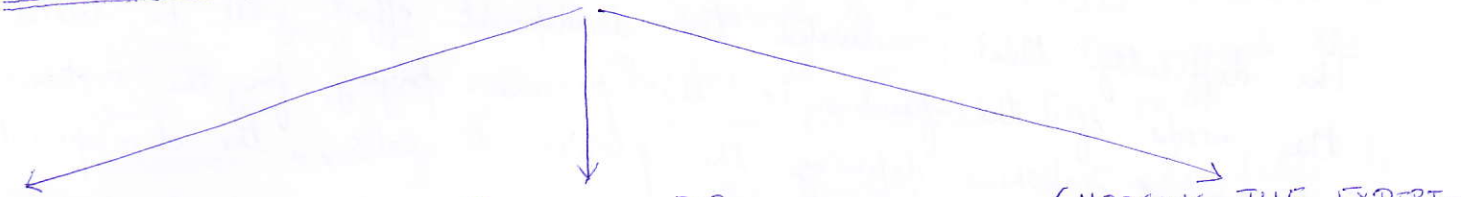
(c) Very difficult: It takes about thirty person-years to build an expert system to solve very difficult problem. e.g. DENDRAL. The time required to build a very difficult problem is between three to five years at a four- to six-person level.




Person-years required to develop an expert system

- The development effort in person-years does not give a very accurate picture of the human resources needed or the time required to build the system.
- The actual time required to build the system depends on problem complexity and the number of full-time professionals assigned to the effort.

# COMMON PITFALLS IN PLANNING AN EXPERT SYSTEM:




## (1) Choosing an Appropriate Problem:

(i) Pitfall: The expert system development effort is addressing a problem so difficult that it can't be solved within the constraints set by the available resources.

How to Avoid it: Have an experienced knowledge engineer develop a small prototype system. Evaluate the results of this effort to decide whether or not to proceed with full-scale development. If decided to proceed, attention is paid to the knowledge engineer's recommendations concerning required resources, problem scope and appropriate tools.

Some problems are appropriate for expert system development while others seem inappropriate. Most problems fall between the two extremes. An experienced knowledge engineer can't predict with certainty whether or not the problem is feasible for expert system work and more analysis is required. This analysis is important because it helps the development team to avoid problems that are inappropriate for the expert system approach.

(ii) Pitfall: The problem that the expert system is designed to solve will not significantly alleviate the difficulty that motivated the development effort.

How to Avoid it: Consider carefully the relationship between the basic difficulty and the target problems i.e. the problems to be solved by the expert system. Will the expert system have the desirable long-term effect if it works perfectly. [illegible] reasonable problem scope, reassess the effectiveness

of the expert system within the new problem scope.

The difficulty that motivated the development effort will be related to the needs of the funders i.e. the persons paying for the system development. When defining the problem to solve, the development team should consider the needs of both the funders and the end-users. If their needs differ or clash, strategies should be found to reconcile them.

(iii) Pitfall: The problem that the expert system addresses is so general or complex that an excessive number of rules and data base objects are needed to describe the expertise adequately. Either it will take too long to accumulate all the rules or the resulting system will run too slowly.

How to Avoid it: If the expert system will require more than a few thousand medium-sized or large rules, reexamine the problem scope to see if it can be constrained.

The problem lies in estimating the number of rules required by the expert system. A small prototype system during the feasibility study is build in order to develop a reasonable estimate. It will solve a simple subproblem in the domain. The size of this system will provide an experienced Knowledge engineer with some idea of the size of the full-blown system.

## (2) Resources for Building the System:

(i) Pitfall: Only a limited amount of time exists in which to build the expert system. Therefore it is decided to provide the personnel necessary to build it in the alloted time period.

How to Avoid it: Don't gear the development effort timetable to production deadlines. Use the recommendations of an experienced Knowledge engineer, one who has performed a feasibility analysis to determine the timetable for development.

Do not speed up expert system development by increasing personnel. The interactive nature of the development process imposes limits on how fast development can take place. A Knowledge engineer who has conducted some preliminary analysis is the best to provide a realistic estimate of development time.

(ii) Pitfall: Management believes that an expert system is a computer program to perform some task and any experienced programming staff can build it. The staff needs is the problem specifications and the right programming environment.

How to Avoid it: The system development should be performed by an experienced Knowledge engineer or Knowledge engineering team. The talent should be brought from outside or personnel should be trained in-house. Use an experienced Knowledge engineer as a consultant to monitor the work and keep it on track.

Many problems can arise from trying to build an expert system with personnel who lack expertise in Knowledge engineering. The system may require unnecessarily long development time, may be difficult to extend or refine and may not be able to explain its operation.

(3) Choosing the Expert-System-Building Tool:

(i) Pitfall: The development team finds it difficult to represent the domain concepts and control structures needed to solve the problem using the chosen tool.

How to Avoid it: The expert-system-building tool to use should be carefully considered. The tool whose characteristics match the solution and system features suggested by the problem domain should be chosen. If development reveals that the tool is inappropriate for the problem, immediately switch to a different tool.

The choice of an appropriate tool for building the system is difficult since the choice depends on characteristics of the problem solution. The development team may not be able to evaluate the chosen tool until a small prototype system has been implemented. Reimplement the system in more appropriate language if the chosen tool appears ineffective for the problem domain.

(ii) Pitfall: The Knowledge engineer picks the most familiar tool even though other tools are better suited to the problem domain. If the tool is ill-suited, the Knowledge engineer reworks the problem so that it fits the capabilities of the chosen tool.

How to Avoid it: Most Knowledge engineers have strong preferences for specific tools. More than one Knowledge engineer should be consulted to develop a consensus (a costly and not entirely satisfactory solution). The Knowledge engineer should be aware of this trap and reexamine the justifications for choosing the tool.

The Knowledge engineer is sometimes more deeply involved with one Knowledge engineering language than with the others. If the Knowledge engineer redefines the problem scope to fit the chosen tool, the resulting expert system will st solve a broad range of problems.

(iii) Pitfall: It is decided to develop the expert system in FORTRAN, PASCAL, C or some other general-purpose programming language because the resulting system will be smaller, faster and more portable. This increases the development time to some unacceptable length.

How to Avoid it: Implement, test and refine the system in a high-level expert system building language. Choose a specialized tool. If the system reaches a point where it performs satisfactorily except for slow execution speed, consider reimplementing the system in a more efficient language.

Knowledge engineering languages such as ROSIE, EMYCIN were designed to speed system development. They provide special features to ease the representation problem and include many support facilities. An expert system can be built much faster in one of these languages. The drawback is that the expert system will execute slowly in these complex Knowledge engineering languages. The time-consuming part of expert system development is extracting the knowledge from the expert and deciding how to represent it.

(iv) Pitfall: The tool chosen to build the expert system is found to contain programming errors that prevent the use of many of the tool's features. The development team must spend a large portion of its time tracking down and correcting errors.

How to Avoid it: An expert-system-building tool that has an established track record and has been used successfully in other application areas is chosen. The new tools that are still under development should be avoided. The tool that is old and no longer maintained or supported should not be chosen.

The development team should consider the status of the tool while choosing it.

# PITFALLS IN DEALING WITH DOMAIN EXPERT :

CHOOSING THE DOMAIN EXPERT

INTERACTING WITH THE EXPERT

## (1) Choosing the Domain Expert :

(i) Pitfall : The Knowledge engineer has great difficulty extracting high-quality rules from the expert. Interactions with the expert are laborious and seem to provide a small payoff.

How to Avoid it : An expert highly skilled in the target domain should be chosen. Rely on the opinions of the expert's peers and other knowledgable people. The expert should be an articulate and a motivator.

The expert should be skilful at finding good solutions to difficult problems. The expert must be open, articulate and able to convey ideas easily to others. The expert must be aware of the advent of computer technology.

(ii) Pitfall : The domain expert chosen for development cannot find enough time for the project.

How to Avoid it : Get a commitment from the expert before the project starts, to spend at least half-time on the project. The expert should understand the importance of his contribution to this effort. It is desirable to pick a nearby expert, preferably in the same city.

The Knowledge engineer builds an expert system through a series of intensive interactions with a domain expert. For this to happen, the expert should be available and accessible i.e. the expert must be able to interact regularly with the Knowledge engineer for extended periods of time and the expert should also be located near the development work.

(2) Interacting with the Expert :

(i) Pitfall : Both the in-house experts and outside experts who interact with the prototype system during development have trouble correcting and modifying the system's rules. They are not sure about the meaning of the rules.

How to Avoid it : The terminology used in the program should be the same that the experts use while solving problems in the domain. All the terms used should be defined and a mechanism should be build into the program that allows the expert to access these definitions. If the program code is not readable, a means for translating the code into statements that the expert can understand should be provided.

Expert system development requires collaboration between the Knowledge engineer and the expert. The expert supplies the Knowledge for the system while the Knowledge engineer organizes and maps the Knowledge into a form usable by the computer. During this mapping process, the Knowledge engineer must provide the system with a translation of the expert's Knowledge. Both the expert and the system must be able to understand the rules embodying the Knowledge.

(ii) Pitfall : The rules generated by the expert are so short and simple that they don't provide a high degree of accuracy when used in complex situations.

How to Avoid it : The expert should be monitored as he solves realistic problems. The problems which differ qualitatively from actual problems in the domain should not be used. The realistic data should be included in the problem statements such as letters, reports, laboratory tests or other data taken from real problems.

The Knowledge engineer must watch the expert solve problems, ask the expert probing questions and develop a set of problems for the expert to solve that span the domain, covering all the types of problems that the expert would encounter in order to extract sufficiently rich information from the domain expert. The Knowledge engineer should systematically vary the data for each problem to explore the solution paths to be taken by the expert. The expert will help to define what constitutes a realistic problem and to identify the kinds of data that must be included. The Knowledge engineer may then create the test problems or rely on other domain experts for assistance.

(iii) Pitfall: The expert no longer seems excited about helping develop the expert system and finds the work somewhat uninteresting and routine. The time that the expert makes available for meeting with the Knowledge engineer is steadily decreasing.

How to Avoid it: The Knowledge engineer and expert should meet frequently on a regular basis at least two or three times a week. The expert should be involved in the actual building and modification of the system. The computer tools should also be provided.

Interview sessions should be scheduled at frequent and regular intervals. Long periods between sessions will reduce the expert's interest and project's momentum. Making the expert feel like an integral part of the system-building process will motivate him or her. Do not ask the expert to try things on the computer that are beyond his or her current abilities.

(iv) Pitfall: The expert is not familiar with computers and doubts that an approach using computers will be of any use.

How to Avoid it: Increase the functionality of the expert system in the workplace gradually.

The experts tend to be skeptical of the academic approach to problem solving and about the value of using artificial intelligence and computers. The experts should be made familiar with the computer in order to test the emerging system.

(v) Pitfall: The Knowledge engineer does not have time to explore the reasoning processes of one or two experts in great depth as so many different experts are being used. The result is a shallow analysis of their problem-solving techniques.

How to Avoid it: Involve just one or two experts at the beginning and interact with these experts until their methods are understood and a working prototype has been built. Additional experts should be used for final testing and revision of the system.

The Knowledge engineer has only a limited amount of time to devote to interviews with experts which leads to a superficial understanding of how an expert solves problems in the domain. Spending substantial amounts of time with very few experts helps to build a strong relationship between them. The Knowledge engineer becomes somewhat of an expert in the problem domain and the expert becomes familiar with AI ideas. A sense of mutual confidence and respect must be developed between the Knowledge engineer and the expert which takes time, patience and exposure.

# PITFALLS DURING THE DEVELOPMENT PROCESS:

SYSTEM IMPLEMENTATION

SYSTEM TESTING AND EVALUATION

## (1) System Implementation:

(i) Pitfall: During system development, the Knowledge of the expert has become so entwined in the program that it is difficult to tell the expert's Knowledge from general problem-solving Knowledge and control or search strategies.

How to Avoid it: The domain-specific Knowledge should be carefully separated from the rest of the program and should be represented in meaningful chunks at a useful level of detail. This is easier if a rule-based organization is used where domain Knowledge is represented as sets of IF-THEN rules.

In expert systems, the Knowledge of the expert is separated from the other Knowledge in the system in order to make the expert Knowledge clear and explicit, simplifying the process of explanation, making easier to extend or augment the system and making it possible to check for inconsistencies in the rules and data.

(ii) Pitfall: The Knowledge engineer has extracted hundreds of rules and has represented them in a high-level Knowledge engineering language after interacting with the expert. However, in implementing the prototype expert system and starting to test it, the Knowledge engineer finds that many fundamental concepts were omitted from the rules.

How to Avoid it: During all phases of development test your ideas by implementing them on the computer and during each phase, revise previously formed ideas.

The evolution of expert system requires cycling back and forth between various phases of development, revising and refining the ideas generated and the experien Knowledge engineer does this in order to save time in the long run.

(iii) Pitfall : The expert system is developed in a language that doesn't provide built-in explanation facilities. After the system performs well, the development team attempts to add mechanisms for allowing the system to explain and justify its operations. The team has difficulty doing this.

How to Avoid it : The system should be designed to facilitate self-examination from the very beginning. The chosen tool should have built-in explanation facilities or simple explanation capabilities should be handcrafted for the early prototypes which will speed implementation. The final system should have a sophisticated explanation facility to assist the end-user.

Self Knowledge helps the expert system analyze its own operation and understand and explain how and why it reaches particular conclusions Standard explanation techniques such as displaying chains of rules used to reach a conclusion are well understood than more sophisticated techniques such as justifying the use of particular rules or strategies. Current explanation facilities speed system development by helping with debugging, testing and refinement and increase user acceptance by inspiring confidence in the system's performance and reasoning processes.

(iv) Pitfall : The expert system contains a very large number of highly specific rules which slows system execution and makes the system complex and unwieldy.

How to Avoid it : Sets of special-purpose rules should be collapsed into single, more general rules which will make the expert system more compact, efficient and manageable. The Knowledge engineer extracts rules from the expert by watching the expert solve actual problems and asking appropriate questions about that problem-solving activity. This tends to produce overly specific or specialized rules. The Knowledge engineer should interact with the expert to gain an understanding of the general principles underlying the rules.

(2) System Testing and Evaluation:

(i) Pitfall: When the expert system is tested and evaluated, the users find its performance disappointing in terms of both quality and utility of answers produced.

How to Avoid it: Start planning evaluation techniques during the identification phase of system building which helps guide system design and assists the development team in assessing the progress being made. During these early phases of development, specify the minimum acceptable performance that will allow the system to be considered a success.

Evaluating an expert system is difficult and in order to insure useful results, the system users should be involved in the system design from the very beginning.

(ii) Pitfall: The users find it difficult to interact with the expert system as they don't understand the error messages, slow response times and keep forgetting how to use the editing system to change or add new rules.

How to Avoid it: Good human-engineering practices should be used in the design of expert system and users should be insulated from technical problems unrelated to the problem domain such as standard editing systems or operating systems. The user should get quick response from the system. The input and output should be clear, concise and stylized form of English.

User acceptance can make or break an expert system. The users will not accept a system with poor performance and the users will also not accept a high-quality performance system which is consuming and tedious. The system should be designed to be sensitive to the needs of different types of users i.e. domain experts,

end-users and clerical staff.

(iii) Pitfall : As the expert system begins to exceed a few hundred rules, corrections or additions to the rules tend to introduce as many as or more errors than they fix and hence debugging seems to be endless.

How to Avoid it : A record of the problems presented to the system and the answers they produce should be maintained and this should be used to create a set of standard problems for testing system consistency. The standard problem set should be run each time whenever major changes or additions to the rules are made so as to look for errors caused by the modifications. Keep track of the rules which produce particular conclusions. The unused rules indicate erroneous rule premises or a faulty control scheme.

An expert system becomes difficult to modify as it grows larger and adding new code may affect the results produced by already existing code. The use of a collection of standard problems to help debug the system can prove invaluable. Run the problem set after system changes are made.