8 CHAPTER

EXPERT SYSTEMS

8.0 INTRODUCTION

Expert Systems are computer programs which store the factual and inferential knowledge of human experts in a narrow domain. They can assist and advise users, provide explanations, inform the user at any point of the solution which hypothesis they are pursuing, why they have chosen a particular strategy and what conclusion they have drawn so far. An expert system is an AI program which uses knowledge to solve problems which would normally require a human expert. It attempts to reproduce the performance of one or more human experts. Let us compare a human expert and an artificial Expert System in a tabular form—

Human Expert	Artificial Expert Systems
1. Humans are naturally perishable.	1. They are permanent.
2. They are expensive.	2. They are costly to develop but cheaper to operate.
3. They are inconsistent.	3. They are consistent.
4. They are difficult to transfer.	4. They are easy to transfer.
5. They are difficult to document..	5. They are easy to document.
6. They are adaptive.	6. They need to be told.
7. They are creative.	7. They are uninspired.
8. They possess sensory experience.	8. They need symbolic input.
9. They have broad focus.	9. They have narrow focus.
10. They can deal with common sense knowledge.	10. They need technical knowledge.

In AI literature, many definitions of expert systems have been given. Let us give some of them now:

1. **Any problem that can be solved by your in-house expert in a 10-30 minute telephone call can be developed as an expert system** (By M. Firebaugh).
2. **Expert system or a KBS includes a sizable KB consisting of facts about the domain and heuristics for applying these facts.**
3. **Expert systems use knowledge rather than data to control the solution process.**
4. **An Expert System is a program that emulates the interaction a user might have with a human expert to solve a problem.**
5. **An expert system is designed to act as an expert in a particular domain.**

But please understand here that an expert system is different from a software system. Let us see how:

Expert System	Software Systems
1. They use knowledge base.	1. They use database.
2. Lesser data is kept together.	2. Large data is kept separately.
3. They use reasoning mechanism and heuristics.	3. They use algorithms.
4. They use inference engine.	4. They use compilers.
5. They are developed by knowledge engineers.	5. They are developed by software engineers.
For example: DENDRAL, MYCIN, LITHO, TIMM, XCON, CLIPS etc.	**For example:** C/C++, JAVA 2 etc.

But we should understand the first point of the above table of difference that knowledge base is different from a database. So, we tabulate the differences in a tabular form now.

Knowledge Base	Data Base
1. It stores information at a higher level of obstraction.	1. They are a collection of data representing facts.
2. It operates on a class of objects.	2. It operates on a single object.
3. They use power of inferencing.	3. Information needs to be explicitly stated.
4. Representation is by logic or rules or frames or scripts or nets.	4. Represented by relational, hierarchical or network model.
5. It is used for data analysis and planning.	5. It is maintained for operational purpose only.

8.1 CHARACTERISTICS OF ES

The main characteristics of ES are as follows—

1. They reduce the cost of accessing information as it allows dissemination of information held by one or small group of experts to less expensive people.
2. Knowledge in an ES can be formalized, tested and validated.
3. They allow integration from different sources.
4. They use symbolic knowledge representations.
5. The response time of an ES should be adequate.
6. Increase the probability, frequency and consistency of making good decision.
7. Help distribute human expertise.
8. Permit dynamism through modularity of structure.

A program talking about 'medicine' will be a KBS whereas a system having knowledge about medicine which is able to perform medical treatment will be an expert system.

8.2 NEED OF AN ES AND ITS JUSTIFICATION

As explained earlier also, a human expert is perishable but artificial expert system is not. It is permanent. If there is one expert then the problem cannot be solved if that expert is not experienced.

So, we take group of experts as then the solution is more efficient. One expert system can be used at different places at the same time. ES are tools of mass production. Emotions also have no effect on expert systems. In nutshell, an ES is like a tutor for someone who is interested in learning. ES have variety of applications—medicine, engineering, geology, I.T., basic and applied research, training, sales and marketing.

8.3 ES ARCHITECTURE

Both factual and heuristic knowledge is required for complex decision. Knowledge engineers design various components which work together to perform different functions. Fig. 8.1 shows the architecture of an ES.

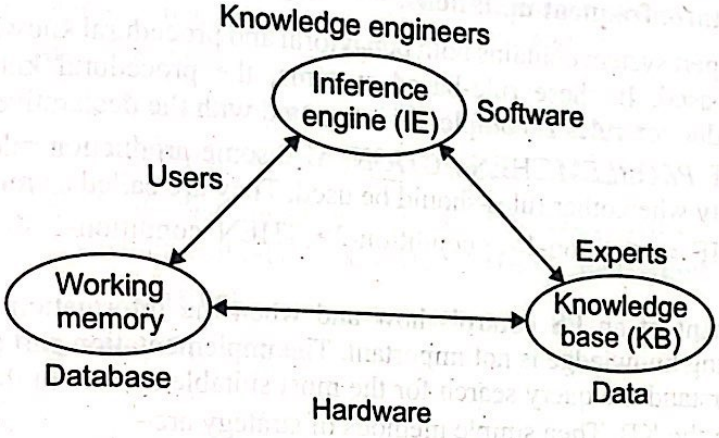

Fig. 8.1 ES architecture.

Knowledge Base (KB): It consists of problem solving rules, procedures and intrinsic data relevant to the problem domain.

Working memory: It refers to task-specific data for the problem under consideration.

Inference Engine (IE): It is a generic control mechanism that applies the axiomatic knowledge in the KB to the task-specific data to arrive at some solution or conclusion.

Do you know? IE such as VP-Expert may come from a commercial vendor.

The KB may be a specific diagnostic KB compiled by a consulting firm and the problem data may be supplied by the end user. **Please note that the KB is the central nucleus of the ES. Also note that a KB is different from a database as a traditional database environment deals with data that have a static relationship between the elements in the problem domain whereas a KB is created by knowledge engineers who translate the knowledge of real human experts into rules and strategies.** These rules can change depending on the existing problem scenario. KB provides the ES with the capability to recommend directions for user inquiry. These three components—KB, IE and working memory—make an ES, a modular design. The **KB constitutes** the problem solving rules, facts etc that a human expert might use in problem solving. The KB is usually stored in terms of if-then rules. The **working memory** represents relevant data for the current problem being solved. The IE is the control mechanism that organizes the problem data and searches through the KB for applicable rules. **A good expert system is expected to grow** as it learns from user feedback. Feedback is fed into the KB to make ES smarter. The three individual components of an ES are dynamic at various levels—

(a) Working Memory (Most dynamic)

The contents of the working memory (or data structure) changes with each problem. So, it is the most dynamic component of an expert system, assuming that it is kept current.

(b) Knowledge Base (Moderately dynamic)

A KB changes only if some new information arises that indicates a change in the problem solution procedure. Changes in KB should be carefully evaluated before being implemented. **Please note that the changes should not be based on just one consultation experience. Also note that a rule that is found to be irrelevant under one problem situation may turn out to be crucial in solving other problems.**

(c) Inference Engine (Least dynamic)

A change is made to the inference engine only if they are required to make inferential engines are changed only if the developer wishes so. **Please understand here that because frequent updates are costlier and disruptive to clients so most commercial software developers try to minimize frequent updations.**

The KB of an expert system contains both behavioral and procedural knowledge. The procedural knowledge is rule-based. In these rule-based systems, the procedural knowledge (in form of IF-THEN ELSE production rules is completely integrated with the declarative knowledge.

For example: IF *PROBLEM* THEN *ACTION*. Also some production rules provide guidelines for which they specify when other rules should be used. They are called as **meta-rules.**

For example: IF <condition-1> <condition-2> THEN condition-1 should be done before condition-2 is done.

The IE component of an ES controls how and when the information in the knowledge is applied. Merely having knowledge is not important. The implementation part of knowledge is most important, i.e., understand the query search for the most suitable answer in the KB and then apply the information from the KB. Then simple methods of strategy are—

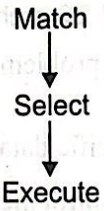

Fig. 8.2

In the process of matching, when the left side of a sequence of rules is initialized first and the rules are executed from left to right, the process is called as **forward chaining. When the right side of rules is instantiated first, the left-hand condition becomes subgoals.** The subgoals may in turn cause sub-subgoals to be established and so on until facts are found to match the lowest subgoals conditions.

The User Interface (UI) component enables us to communicate with an expert system. The **communication performed is bi-directional**. This is shown in Fig. 8.3.

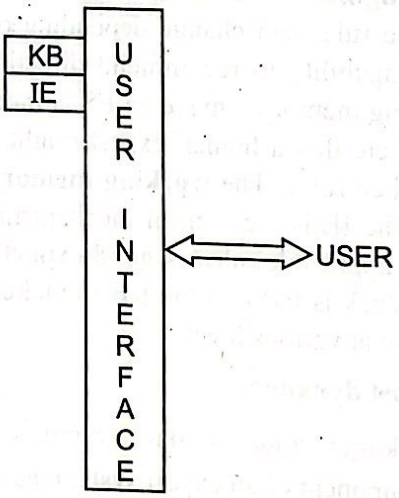

Fig. 8.3 ES components.

As the users are not as experienced as the experts, so the expert systems should be so developed that it has an easy access for the user.

8.4 STEPS TO DEVELOP AN ES

A **knowledge engineer** or an **AI specialist** or a programmer is skilled in the art of developing Expert Systems (ES).

A domain expert is an individual who has significant expertise in the domain of the ES being developed. **Please understand that it is not critical that the domain expert understand AI or expert systems that is one of the functions of the knowledge engineer. Also note that the knowledge engineer and the domain expert usually work close to each other throughout the development process.** An expert system (ES) is developed and refined over a period of several years. ES development has five stages—

S1: Determining the characteristics of the problem **(Identification).**

S2: Finding the concept to produce the solution **(Conceptualization).**

S3: Designing structures to organize the knowledge **(Formalization).**

S4: Formulating rules which embody the knowledge **(Implementation)**

S5: Validating the rules **(Testing).**

Development of an ES involves five steps—

Step 1: Identification.

Determining the characteristic of the problem.

Step 2: Conceptualization.

Here, the iterative process between the knowledge engineer and the domain expert continues. Diagrams are often used to clarify the problem. During this process of conceptualization it is sometimes necessary to revise the description that is generated in the stage of identification. The basic aim is to develop a clear concept of the problem.

First step of identification is shown in a flowchart form also—

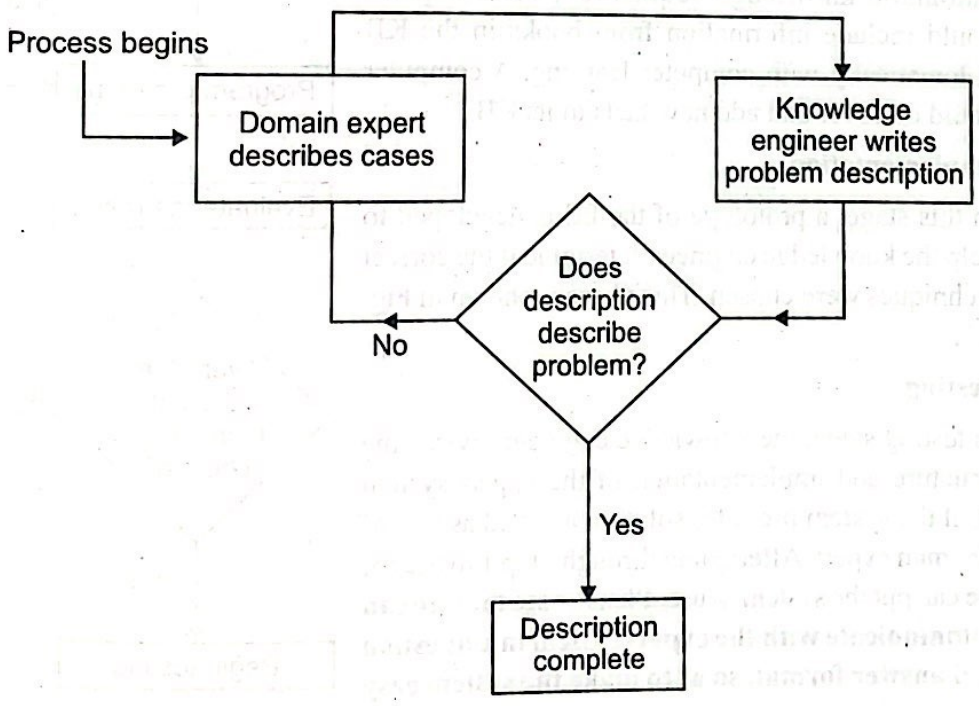

Fig. 8.3 Iterative process of identifying the problem–ES is to solve.

Next is the conceptualization phase. The iterative relationship between the identification and conceptualisations stages of ES development is shown next:

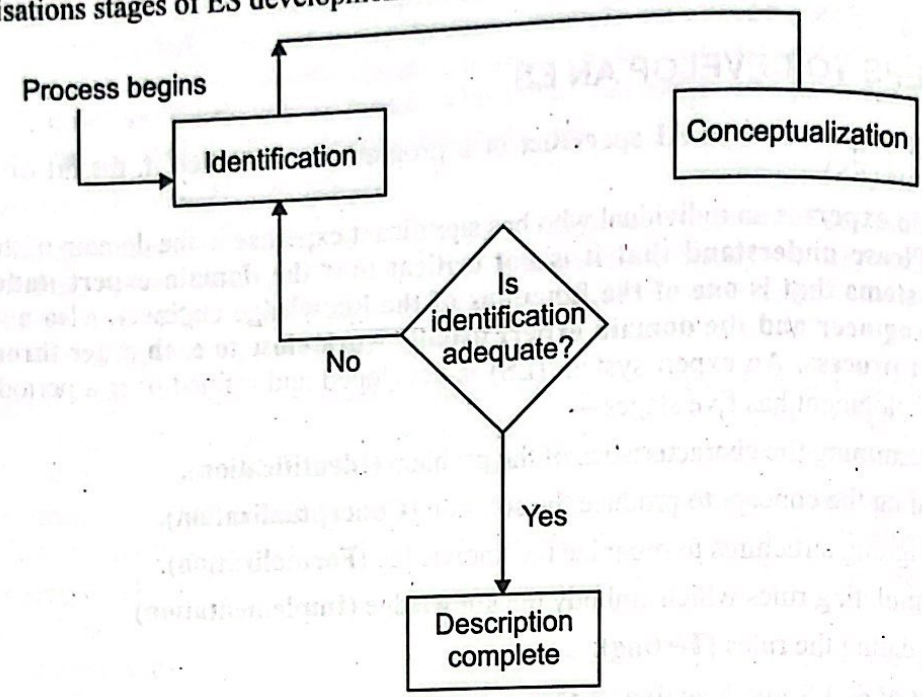

Fig. 8.4 Iterative relationship between identification and conceptualization stages.

Step 3: Formalization

In this step, the knowledge engineers select the development techniques to be used for the expert systems. If a rule-based system is developed, the knowledge engineer develops a set of rules for the domain expert to review until everyone agrees on them. To facilitate the formalization state, the AI researchers are looking for ways to reduce the amounting time required to enter the information in the KB. The goal of the simpler development tools is to enable the domain expert to create the KB. With automatic knowledge acquisition, the computer could include information from books in the KB automatically with computer learning. A computer could discover and add new facts to its KB.

Step 4: Implementation

In this stage, a prototype of the ES is developed to help the knowledge engineer determine if the correct techniques were chosen. This phase is shown in Fig. 8.5.

Step 5: Testing

In testing stage, the knowledge engineer revises the structure and implementation of the expert system until the system provides solution as valid as that of a human expert. After going through Step 1 to Step 5, we can put the system to use. **Please note that we can communicate with the expert system in a question and answer format, so as to make the system easy to use.**

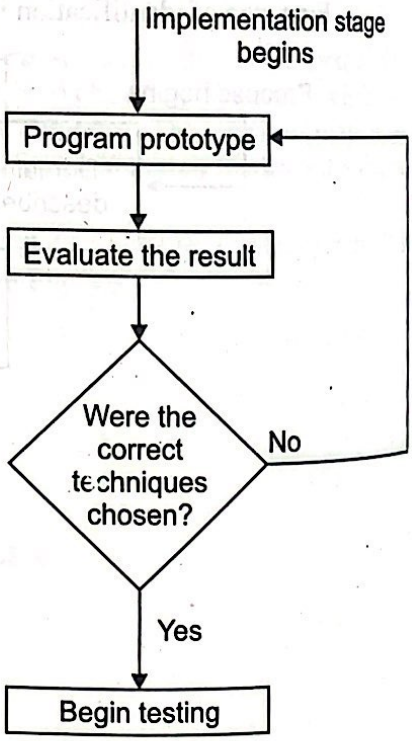

Fig. 8.5 Implementation stage.

The chances of prototype expert system executing flawless are very less initially. A **knowledge engineer** does not expect the testing process to verify that the system has been constructed entirely correctly. Testing provides an opportunity to identify the **weaknesses in the structure** and the **implementation of the system.** Testing includes—

(*a*) **System** was implemented correctly or incorrectly.

(*b*) **Rules** were implemented correctly or not.

Also understand that the results from the tests are used as feedback to return to the previous stage and adjust the performance of the system.

(*c*) System is tested for both **simple and complex problems** by the domain experts to uncover more of defects.

(*d*) An ES is finally tested to be successful only when it operates at the level of a human expert.

(*e*) The testing process is NOT complete until it indicates that the solutions suggested by the expert system are consistently valid (as valid as those provided by a human domain expert).

(*f*) The **ultimate test** of an expert system is how well it performs, not in a lab but in real-life situations. **Please understand that a completed system not only must demonstrate consistently that it can deliver the required expertise but it must also be easy to use, so that the embedded expertise can be extracted easily.** That is why, developers have gone to the extent that expert systems are among the easiest to use of all sophisticated programs.

(*g*) Expert systems are typically **interactive,** *i.e.*, they work in question and answer form. This interaction between users and ES continues until the system is able to reach a conclusion (displayed on to the screen).

8.5 CASE STUDIES: MYCIN, DENDRAL, RI AND OTHERS

Let us discuss some expert systems now.

MYCIN

MYCIN is an Expert System developed in 1970s at Stanford University. It was written in LISP language as the part of doctoral dissertation of Edward Shortliffe. Its job was to diagnose and recommend treatment for certain blood infections. It was developed to assist a physician who is not an expert in the field of antibiotics for the treatment of blood infections.

MYCIN'S KB is organized around a set of rules of the form mentioned below:

"IF condition-1 and .....condition-n are true then draw conclusion-1 and ..... conclusion n".

For example:

IF
The infection is primary bacteremia.
AND the site of culture is one of the sterile sites
AND the suspected portal of entry is gastrointestinal tract
THEN
there is suggestive evidence (0.7) that infection is bacterial.

Here, the number 0.7 is called as the **Certainty Factor** (CF), which indicates how certain the conclusion is, if given conditions are satisfied.

The value of CF indicates the amount of truthness of conclusion. It states that— **"If condition-1 holds with certainty xl.....and condition-m holds with certainty-xm then draw conclusion-1 with certainty y1 and conclusion-n with certainty-yn. When we have multiple certainty factors, then their combined CF is the minimum of individual CFs AND CF (action) = CF (Premise) * CF (rule).**

How MYCIN works?

MYCIN adopts a high-level approach for the task of consultation , which is mentioned below—

If

there is a patient which requires therapy AND Consultation has been given to other organisms requiring therapy

Then

Compile a list of possible therapies and determine best one.

Herein, the symptoms of patient are matched with premise and if matching occurs, **the rule is fired.** To find out whether some rule is applicable or not, MYCIN first finds out, if there is indeed an organism present which is associated with significant disease. **Please understand that this information is obtained directly by user or via a chain of inferences. The consultation is search through a tree of goals.** The top goal is at the root of the tree and then it is divided into subgoals.

Do you know? MYCIN's control structure uses AND/OR tree.

During disease findings, sometimes multiple answers are found. To find the correct and actual output, the order in which facts and rules appears is checked. This corresponds to depth first search.

MYCIN answers queries of type "**Why**" and "**how**".

1. 'Why' questions can be asked at any time from the system. The system then responds with the action part of the rule. It displays 'how' question tells the reasoning process of how the answer was found.

Many other variants of MYCIN project are there like—

(*a*) EMYCIN: First expert shell developed from mycin'.

(*b*) PUFF: Developed using EMYCIN in the new domain of heart disorders.

(*c*) NEOMYCIN: Developed to train doctors which would take them through various example cases, checking their conclusions.

DENDRAL

The word "Dendral" is a pruned version of "Dendritic Algorithm". It was designed and developed at Stanford University in 1965. It is a collaborative work between Edward Feigenbaum, Bruce Buchanan, Joshua Lederberg and Carl Djerassi. **DENDRAL is an expert system developed for inferring process of structure elucidation of chemical compounds.** The **primary** aim was to aid organic chemists with the identification of unknown organic molecules by analyzing information from mass spectrometry graphs and the knowledge of chemistry. Dendral is not just a single program but rather a product of an interaction between two subprograms—**Heuristic-Dendral** and **Meta-Dendral,** i.e.,

DENDRAL = Heuristic-Dendral + Meta-Dendral

It is written in LISP language. Let us first of all understand these two subprograms.

Heuristic Dendral is a program that uses mass spectra or other experimental data together with knowledge base of chemistry, to produce a set of possible chemical structures that may be responsible for producing the data. A mass spectrum of a compound is produced by a mass spectrometer, and is used to determine its molecular weight, the sum of the masses of its atomic

constituents. **For example**, the compound water (H_2O) has a molecular weight of 18 since hydrogen has a mass of 1.01 and oxygen 16.00, and its mass spectrum has a peak at 18 units. Heuristic Dendral would use this input mass and the knowledge of atomic mass numbers and valence rules, to determine the possible combinations of atomic constituents whose mass would add up to 18. As the weight increases and the molecules become more complex, the number of possible compounds increases drastically. Thus, a program that is able to reduce this number of candidate solutions through the process of hypothesis formation is essential.

Meta-dendral:

Meta-dendral is a knowledge acquisition system that receives the set of possible chemical structures and corresponding mass spectra as input, and proposes a set of hypotheses to explain correlation between some of the proposed structures and the mass spectrum. These hypotheses would be fed back to Heuristic dendral to test that applicability. Thus, "Heuristic dendral is a performance system and Meta-dendral is a learning system". The program is based on two important features: the plan-generate-test paradigm and knowledge engineering.

DENDRAL uses the control process of generate and test procedure (like state space search), starting with arbitrary initial state and by employing a generator, all possibilities regarding presence of compounds is generated. On these possibilities a series of tests are carried out to eliminate unwanted states.

The plan-generate-test paradigm is the basic organization of the problem-solving method, and is a common paradigm used by both **Heuristic Dendral** and **Meta-Dendral** systems. The **generator** generates potential solutions for a particular problem, which are then expressed as chemical graphs in Dendral. However, this is feasible only when the number of candidate solutions is **minimal**. When there are large numbers of possible solutions, Dendral has to find a way to put constraints that rules out large sets of candidate solutions. This is the primary aim of Dendral **planner**, which is a "hypothesis-formation" program that employs "task-specific knowledge to find constraints for the generator". Last but not least, the **tester analyzes each proposed candidate solution and discards those that fail to fulfil certain criteria. This mechanism of plan-generate-test paradigm is what holds Dendral together.**

CONGEN (CONnstraints GENerator) is a Dendral program which constructs complete chemical structures by manipulating symbols that stand for atoms and molecules. Its input is a molecular formula and a set of constraints which serve to restrict the possible inter-connections among atoms. As an output, it generates a list of all possible ways of assembling the atoms into molecular structure by imposing the given constraints.

The KB of DENDRAL basically consists of huge knowledge about organic chemistry. The entire knowledge about organic chemistry is coded in the form of production rules. One example of such a rule is—

If there are two peaks at mass units $m1$ and $m2$

AND $x1 + x2$ = molecular weight + 28

AND at least one of $x1$ or $x2$ is high

Then ketone group is present.

How DENDRAL works?

Initially, **compound testing** is planned. It is fed into program and various groups are identified. The planner's production rule MSPRUNE is used to prune the number of alternate structures in the compound. The **major task** of MSPRUNE is to generate a hypothetical spectrum for structure. To perform this task it requires knowledge about characteristics spectra of all organic groups. **This knowledge is coded in form of production rules.** Another module, MSRANK, ranks the candidates into various groups. This subsystem has the knowledge of spectrometry to rank candidate structure.

DENDRAL has the following organizational features:

1. **Knowledge Representation:** Production rules and algorithms for generating graph structures are constructed by META DENDRAL— a program which uses learning techniques to construct rules for an expert system automatically.
2. **Reasoning:** Forward chaining (data driven).
3. **Heuristics:** DENDRAL uses a variation of dfs called as generate and test. In this all hypotheses are generated and then tested against available evidence. Heuristic knowledge from the users (chemists) is also used to constrain the search.
4. **Dialogue/explanation:** The dialogue uses mixed control. The user can supply information and the system can request information as required.

Do you know? Many systems were derived from Dendral like MOLGEN, MACSYMA, PROSPECTOR, XCON and STEAMER.

R1/XCON

The R1 (later called as XCON—expert CONfigurer) program was a production-rule-based system written in OPS5 by John P. McDermott of CMU in 1978. It helped in assisting of the ordering of DEC's VAX computer systems by automatically selecting the computer system components based on the customer's requirements.

XCON interacts with the sales person, asking critical questions before printing out a coherent and workable system specification. Let us summarise R1/Xcon/features—

1. John McDemott (1982–84) of Carnegi-Mellon University, USA, wrote R1/XCON in collaboration with DEC (Digital Equipment Corporation).
2. XCON is an acronym for **expert configurer** and is used to configure VAX and PDP-11 series of computer systems sold by DEC.
3. **XCON name is used by DEC; R1 name is used by McDermott.**
4. Initially, the system contained about **300 production rules** of the standard variety for configuring the **420 possible** components of a VAX–11 system.
5. A completed system containing some 800 production rules was delivered to DEC for acceptance testing (Oct. 1979). By 1980, DEC personnel had become sufficiently familiar with the operation of XCON to assume full responsibility for its maintenance and development and they have extended it to contain over **4000 rules.**
6. XCON system was designed to replace human **"technical editors".** Such editors had been responsible for 2 operations—
   (*a*) Checking for completeness of orders,
   (*b*) Laying out the physical placement and wiring connections between component modules in the chassis.

   These tasks are fairly tedious and each of the components is specified by properties such as **voltage** (*V*), **frequency** (*f*), how many **devices** it supports and how many **parts** it has. Also, the editors had information showing which components can or must be associated with other components. **This set of rules governing the relationship between components is called as constraint knowledge.**
7. XCON passed an extensive series of **performance tests.**

NOTE: To text XCON, a set of 10 configurations was submitted to the program. Results were evaluated by a team of 12 people including technical editors, technicians and engineers. Errors detected were corrected and the cycle was repeated until 50 configurations had been produced.

8. By 1986, XCON represented an investment of over 50 person-years of human effort. Since its birth, XCON has grown at a relatively uniform rate as its capabilities expanded from a single line of VAX computers to the complete VAX line and the PDP–11 line. This required an effort of about four person-years per year.
9. XCON developers found that they did not have to complete the system before putting it to use. This shows XCON's **incremental growth.**
10. XCON's developers have concluded that it will never be possible **to guarantee 100% error-free performance.** Also they observed that building an ES is an unending process.
11. Since 1980, XCON has analyzed over 100,000 unique orders and is presently operating with greater than 99% accuracy. So, it is an effective business tool today.

SUMMARY

In this chapter, we have defined what is an expert system. Why are they required? What is its architecture? How to develop an ES? Some case studies on MYCIN, DENDRAL, RI (or XCON) have been discussed.

MULTIPLE CHOICE QUESTIONS [MCQS]

Q.1. The class of program first developed by AI researchers is
(*a*) Human expert system (*b*) Artificial expert system
(*c*) Automatic expert system (*d*) None of the above.

Q.2. ES uses
(*a*) Knowledge (*b*) Data (*c*) Information (*d*) None of the above.

Q.3. Data when processed becomes information and information when processed becomes
(*a*) Data (*b*) Information (*c*) Knowledge (*d*) Wisdom.

Q.4. Which of the following is an example of an ES?
(*a*) C (*b*) C++ (*c*) JAVA 2 (*d*) MYCIN.

Q.5. Which of the following is an exmaple of an IE?
(*a*) VP-Expert (*b*) VIP-Expert (*c*) D-Expert (*d*) C-Expert.

Q.6. DENDRAL has two subprograms, namely
(*a*) Heuristic Dendral + KB (*b*) Heuristic Dendral—Meta Dendral
(*c*) Heuristic Dendral + Meta Dendral (*d*) None of the above.

Q.7. DENDRAL uses
(*a*) MPRUNE (*b*) MS PRUNE (*c*) SPRUNE (*d*) None of the above.

Q.8. XCON stands for
(*a*) Expert configurer (*b*) Extra configuration
(*c*) E-console (*d*) None of the above.

Q. 9. To train doctors, which ES was developed?
(*a*) EMYCIN (*b*) PUFF (*c*) NEDMYCIN (*d*) None of the above.

Q. 10. An example of an ES shell is
(*a*) Crystal (*b*) C (*c*) LISP (*d*) None of the above.

ANSWERS

1. (*b*)	**2.** (*a*)	**3.** (*c*)	**4.** (*d*)	**5.** (*a*)
6. (*c*)	**7.** (*b*)	**8.** (*a*)	**9.** (*c*)	**10.** (*a*)

CONCEPTUAL SHORT QUESTIONS WITH ANSWERS

Q. 1. How do we use Domain exploration, Meta knowledge and Expertise transfer in building expert system? **[GGSIPU, B. Tech (CSE) 8th Sem., April 2011]**

Ans. **Domain knowledge** is that knowledge that software programs encode in form of different knowledge bases. Domain knowledge is important, because it usually must be learned from software users in the domain (as domain specialists/experts), rather than from software people. Communication between end-users and software people is often difficult. They must find a common language to communicate.

Meta knowledge or **meta-knowledge** is knowledge about knowledge. More precisely speaking, meta-knowledge is systematic problem and domain-independent knowledge which performs or enables operations on another more or less specific domain-dependent knowledge in different domains/areas of human activities. Meta-knowledge can be considered as a fundamental conceptual instrument in such research and scientific domains as, knowledge engineering, knowledge management, and others dealing with study and operations on knowledge, seen as a unified object/entities, abstracted from local conceptualizations and terminologies.

Expertise transfer discusses technology and organizational behaviour issues that are barriers to the commercial success of expert systems. The issues include interfacing and integrating with existing systems, picking the correct problem, fulfilling development, goals, representing knowledge, acquiring knowledge, validating and maintaining the system, deploying and introducing expert systems in an organization, and managing organizational behaviour after development is complete. It is argued that overcoming these barriers requires technology-transfer methods that account for an expert system's size and the structure and culture of the organization involved.

Q. 2. Distinguish between DENDRAL and MYCIN in a tabular form.

Ans.

DENDRAL	MYCIN
1. It uses production rules and algorithms for generating graph structures.	1. It uses production rules implemented LISP.
2. Reasoning is done using forward chaining.	2. Reasoning is done using backward chaining.
3. It uses **data-driven** approach.	3. It uses **goal-driven** approach.
4. It uses Generate-and-test strategy.	4. It uses depth-first search.
5. It uses mixed control dialogue explanation.	5. It uses computer control for dialogue/ explanation.

Q. 3. What is the difference between an active user and a passive user?

Ans. An **active user** uses an ES directly to obtain recommendations.

A **passive user** is one who trusts the results obtained from ESs and supports the implementation of those results.

Q. 4. What is heuristic reasoning? Discuss search control methods. Distinguish between forward and backward chaining.

Ans. Human experts use a type of problem solving technique called as heuristic reasoning. Also called as the rules of thumb, it allows the expert to arrive at a good solution quickly and efficiently.

The inferencing (done by IE) is of two types—

(*a*) Forward chaining

(*b*) Backward chaining.

Forward Chaining (or data driven search)

Also called as **data driven search or antecedent search. The process of inferencing that starts with the available data and uses the inference rules to conclude more data until the desired goal is reached is called as forward chaining. An inference engine** using forward chaining searches the inference rules until it finds one in which the if-clause is known to be true. It then concludes the then-clause and adds this information to its data. It would continue to do this until a goal is reached. Because the data available determines which inference rules are used, so this method is also called as **data-driven search.**

How it is done? The given condition (fact) is matched with the left part of a rule (called as **antecedent**). When the antecedent conditions are met, **the rule is fired.** That means the rule is replaced by RHS of the rule. It now becomes the new fact to be matched with the antecedent part of another rule.

Backward Chaining (or goal-driven search)

The process of inferencing that starts with a list of goals and works backwards to see if there is data which will allow it to conclude any of these goals is called as backward chaining. An inference engine using backward chaining would search the inference rules until it finds one which has a **then-clause** that matches a desired goal. If the if-clause of that inference rule is not known to be true then it is added to the list of goals. It is a reverse process of inferencing.

How it is done?

In this, rule interpreter starts with matching "THEN" part of the rule and if match occurs, it is replaced by "IF" condition. Here, the rule interpreter starts with goal state and proceeds towards start state.

Do you know? Situations where the starting states are more than goal states, use forward inferencing.

Q. 5. Consider the set of facts F1, F2, F3.....F6 and set of consequents C1, C2, C3.....C5. The production rules are as follows—

Rule 1: IF F1 and F2 then C1

Rule 2: IF F3 and C1 then C3

Rule 3: IF F6 then C3

Rule 4: IF F2 and C3 then C4

Rule 5: IF C4 then C5

Rule 6: If C2 then C3

Rule 7: If C5 then C6

Assume that input is facts F1 and F3 that are true and the query is that whether consequent C6 is true or not. Use forward chaining and prove this query. Also use backward chaining, if user specifies that C6 has to be proved. Assume that F2 and F3 are true again. Prove that A1 is true.

Ans. (*a*) **Using forward chaining**

Rule applied	Rule States	Inferred fact
1.	applied (fired)	C1
2.	applied (fired)	C3
3.	not required	
4.	applied (fired)	C4
5.	applied (fired)	C5
6.	not required	
7.	applied (fired)	C6

Thus, rules 1, 2, 4, 5, 7 are fired. The inferred fact becomes C6 which is proved to be true.

(*b*) Using backward chaining

If user specifies that C6 has to be proved. Let us assume that F2 and F3 are true again. The inferencing is done as follows—

	Rule Applied	Condition matched	Inferred
C6 (given as true)	7		C5
C5	6, 5	Condition matched	C2, C4
C4	4	Condition matched	C2 and A2
C3	3	Condition matched	A3
C2	2	Condition matched	A3, C1
C1	1	Condition matched	A1, A2

With this sequence of steps, A1 is proved to be true.

Q. 6. Contrast and compare expert systems and neural nets.

Ans. The current developments in ANN technologies present new opportunities for expert system work. The following points explain this—

1. Expert system elements may be implemented with NN.
2. Areas such as speech and vision are done better with NNs.
3. ESs can be used for training NNs.
4. Knowledge acquisition and knowledge engineering problems can be addressed using NN.

5. Implicit knowledge may be provided by NNs to support explicit rule-based knowledge.

Q. 7. Write a short note on ES tools. **[GGSIPU, B. Tech. (CSE), 8th Sem., 2010]**

Ans. The commercialization of AI has been mainly in the form of expert systems and knowledge based systems. In the decade of 1980s expert systems have progressed from efforts in research laboratories aimed at building stand-alone one-of-a-kind systems, to products built and deployed in large numbers in industrial laboratory and government office settings. When early expert systems were constructed, their implementers typically began with a general purpose language such as LISP and built a number of utility data structures and programs to represent facts, rules and strategies and to perform the basic operations of the system such as rule invocation, matching, computations involving uncertainties, etc. Initially, it was often very difficult to speculate now much of that machinery was problem-specific and how much would be reusable for other fields. Once a few experimental applications had been built, the developers begin the task of abstracting out of these programs those aspects that appeared universal. Thus the well known first generation expert system MYCIN led to the development of the tool (or shell) EMYCIN (empty MYCIN). EMYCIN thus provides, based on the past experience, with diagnosis tasks associated with the diagnosis of infectious diseases, a tool with useful knowledge organization and appropriate problem solving methods for generic tasks such as medical diagnosis, pattern classification and equipment fault diagnosis.

EXPERT SYSTEM TOOLS (also called knowledge Engineering Tools) are software systems that support the development, execution and maintenance of expert systems (or knowledge based systems). As a result of widespread interest in expert system development, there has been a huge volume of activity in expert system tool development with market projections of the order of US $ 400 million by early 1990s. The tools also range for large development environments such as ART (Automated Reasoning Tool), KES (Knowledge Engineering System), CLIPS (the C Language Integrated Production System) Level 5, VAS OPS5, etc. The widely varying costs of these packages together with the often exaggerated claims in vendor literature makes selection of expert system tools a difficult task.

Table below gives a partial list of some of these tools—

Advisor	FranzLisp	Personal Consultant Plus
ART	GoldenCommonLisp	PROLOG2
ART-IM	GoldWorks	RuleMaster
Cambridge LISP68000	Insight2+	RuleMaster2
CLIPS	KEE	S.1
ExperLisp	KES	Smalltalk
Experops5	Level 15	Smalltalk
ExpertEase	MicroExpert	VAXOPS5
Expert4	NEXPERTOBJECT	ZEXPERT

Q. 8. Explain the knowledge acquisition methods adopted by expert systems.

[RTM Nagpur Univ., BE (IT)- 7th Sem., Summar 2009]

Ans. Knowledge acquisition is the process of gathering knowledge. The knowledge engineers do this by interacting and interviewing the human experts of that domain. The person interviewing the expert should also have some basic knowledge of the concerned field to find out what to ask. This collected knowledge is then encoded in a format suitable for storage in memory. This job of encoding is done by a computer specialist.

For example: COMPASS is an expert system which performs knowledge acquisition by interviewing.

The knowledge acquisition cycle performs the following steps—

(*a*) Acquires knowledge from expert.

(*b*) Documents the acquired knowledge.

(*c*) Tests the new knowledge.

Q. 9. Discuss some explanation facilities as used by ES.

Ans. Almost all expert systems can explain to users how they reach particular conclusions. The most common type of explanation mechanism deals with the following:

1. **Retrospective Reasoning**

   It explains how the system reached a particular state. For example, the end-user may wish to know why the system needed the answer to the question it just asked or how the system arrived at a certain conclusion. Here, the system may describe the rules which led to the question or display part of the chain or sequence of rules that led to the conclusion.

2. **Hypothetical Reasoning**

   The system explains what would have happened differently if a particular fact or rule had been different.

3. **Counterfactual Reasoning**

   The system explains why an expected conclusion was not reached.

EXERCISE QUESTIONS

Q. 1. Explain different components of an expert system.

[RTM Nagpur Univ., BE (IT)-7th Sem., Summer 2009]

Q. 2. (*a*) Explain rule-based architecture of expert system.

(*b*) What are different applications of expert system?

(*c*) What do you mean by expert system shell?

[RTM Nagpur Univ., BE (IT)-7th Sem., Winter 2009]

Q. 3. (*a*) Draw and explain architecture of Expert System.

(*b*) What is Expert System Shell? State its advantages.

[RTM Nagpur, BE (CSE)-7th Sem., Summer 2009]

Q. 4. (*a*) What are the various methods of knowledge acquisition used in expert system? Explain with examples.

(*b*) What are various knowledge representation techniques used in expert systems? Explain with examples.

[PTU., M.C.A-4th Sem., 2009]

Q. 5. (*a*) What do you mean by an Expert System? Why we need an Expert System? What are different kinds of problems faced during development of an Expert System?

(*b*) Identify and describe two good application areas for expert systems within a university system.

[PTU., B. Tech (CS)-7th Sem., 2009]

Q. 6. (*a*) What is an ES? Explain its various parts.

(*b*) Explain, in short, the working of DENDRAL and MYCIN.

(c) Discuss the concept of uncertainty in ES.

[UPTU., B. Tech (CSE)-8th Sem., 2006-07]

Q. 7. Briefly explain any of the expert system RI (XCON) or Mycin.

[GGSIPU, B.Tech (CS)-8th Sem., May 2008]

Q. 8. Explain the architecture of an expert with a diagram.

[GGSIPU, B.Tech. (CSE)-8th Sem. May-June 2009]

Q. 9. What procedure is followed for knowledge acquisition?

[GGSIPU., B. Tech (CSE)-8th Sem., May-June 2010]

Q. 10. What are the various steps for development of Expert Systems?

[GGSIPU., B. Tech (CSE)-8th Sem., May-June 2010]

Q. 11. How is an Expert System different from other softwares like DBMS etc.

[GGSIPU., B. Tech (CSE)-8th Sem., May 2011]