NATIONAL OPEN UNIVERSITY OF NIGERIA

SCHOOL OF SCIENCE AND TECHNOLOGY

COURSE CODE: CIT 474

COURSE TITLE: INTRODUCTION TO EXPERT SYSTEMS

Course Code	CIT 474
Course Title	Introduction to Expert Systems
Course Developer/Writer	Olayanju Taiwo Abolaji Computer Department, Federal College of Education (Tech.) Akoka, Lagos
Course Editor	
Programme Leader	
Course Coordinator	

NATIONAL OPEN UNIVERSITY NIGERIA

CONTENTS **PAGE**

COURSE GUIDE

Introduction

Introduction to Expert Systems is a First Semester course. It is a two credit degree course available to all four hundred level students offering Information Technology. The course consists of 15 units necessary to acquaint you with the basics of Expert Systems. which will enable you to develop the skills necessary to develop, operate and maintain software. They are no compulsory pre-requisites to it, although it is good to have a basic knowledge of operating a compute system.

What You will Learn in this Course

This Course consists of units and a course guide. This course guide tells you briefly what the course is about, what course materials you will be using and how you can work with these materials. In addition, it advocates some general guidelines for the amount of time you are likely to spend on each unit of the course in order to complete it successfully.

It gives you guidance in respect of your Tutor-Marked Assignment which will be made available in the assessment available. There will be regular tutorial classes that are related to the course. It is advisable for you to attend these tutorial sessions. The course will prepare you for challenges you will encounter in the field of software engineering.

Course Aims

The aim of the course is simple. The course aims to provide you with an understanding of Expert Systems.

Course Objectives

To achieve the aims set out, the course has a set of objectives which are included at the beginning of the unit. You should read these objectives before you study the unit. You may wish to refer to them during your study to check on your progress. You should always look at the objectives after completion of each unit. By doing so, you would have followed the instruction in the unit. Below are the comprehensive objectives of the course as a whole. By meeting these objectives, you should have achieved the aims of the course as a whole. In addition to the aims above, this course sets to achieve some objectives. Thus, after going through the course, you be able to:

· Basic Concept of Expert Systems
· Components, History and the Development of Expert Systems
· Expert Systems Characteristics and Application
· Natural Language interface for Expert Systems
· Programming Language for Developing Expert System
· A rule-based expert system
· Blackboard Expert System – HEARSAY
· Frame Based Expert System- KEE
· Expert System Shells
· A Group Proposal Submitted on Expert System.

Working through this Course

To complete this course, you are required to read each study unit, read the textbook and read other materials that may be provided by the National Open University of Nigeria. Each unit contains self-assessment exercises and at certain point in the course, you will be required to assignments for assessment purposes. At the end of the course there is a final examination. The course should take you about a total of 17 weeks to complete. Below you will find listed all the components of the course, what you have to do and how you should allocate your time to each unit in order to complete the course on time and successfully. This course entails that you spend a lot of time to read. I would advice that you avail yourself the opportunity of attending the tutorial sessions where you have the opportunity of comparing your knowledge with that of other people.

The Course Materials

The main components of the course are:

· The course Guide
· Study Units
· References/Further Readings
· Assignments
· Presentation Schedule

Study Unit

The study units in this course are as follows:

MODULE 1 Basic Concept of Expert Systems
Unit 1 What are Expert Systems?
Unit 2 Components, History and the Development of Expert Systems
Unit 3 Expert Systems Characteristics and Application
Unit 4 Natural Language interface for Expert Systems
Unit 5 Programming Language for Developing Expert System

MODULE 2 Classes of Expert System
Unit 1 A rule-based expert system
Unit 2 Blackboard Expert System – HEARSAY
Unit 3 Frame Based Expert System- KEE
Unit 4 Expert System Shells

Module 3 A Group Proposal Submitted on Expert System.

Each unit consists of one or two week's work and includes an introduction, objectives, reading materials, conclusion, and summary, Tutor Marked Assignment (TMAs), references and other resources. The unit directs you to work on exercises related to the required reading. In general, these exercises test you on the materials you have just covered or required you to apply it in some way and thereby assist you to evaluate your progress and to reinforce your comprehension of the material. In addition to TMAs, these exercises will help you in achieving the stated learning objectives of rhe individual units and of the course as a whole.

Presentation Schedule
Your course materials have important dates for the early and timely completion and submission of your TMAs and attending tutorials. You should remember that you are required to submit all your assignments by the stipulated time and date. You should guard against falling behind in your work.

Assessment
There are three aspects to the assessment of the course. First is made up of self assessment exercises, second consists of the Tutor_Marked Assigment and third is the written examination/end of course examination. You are advised to do the exercises. In tackling the assignments, you are expected to apply information, knowledge and techniques you gathered during the course. The assignments must be submitted to your facilitator for formal assessments in accordance with the deadlines stated in the presentation schedule and the assignment file. The work you submit to your tutor for assessment will count for 30% of your total course work. At the end of the course you will need to sit for a final or end of course examination of about three hour duration. This examination will count for 70% of your total course mark.

Tutor-Marked Assignment
The TMA is a continuous assessment component of your course. It accounts for 30 % of the total score. You will be given four (4) TMAs to answer. Three of these must be answered before you are allowed to sit for the end of course examination. The TMAs would be given to you by your facilitator and returned after you have done the assignment. Assignment questions for the units in this course are contained in the assignment file. You will be able to complete your assignment from the information and the material contained in your reading, references and the study units. However, it is desirable in all degree level of education to demonstrate that you have read and researched more into your references, which will give you a wider view point and may provide you with a deeper understanding of the subject. Make sure that each assignment reaches your facilitator on or before the deadline given in the presentation schedule and assignment file. If for any reason you can not complete your work on time, contact your facilitator before the assignment is due to discuss the possibility of an extension. Extension will not be granted after the due date unless there are exceptional circumstances.

Final Examination and Grading
The end of your examination for Introduction to Expert Systems will be for about 3 hours and it has a value of 70% of the total course work. The examination will consist of questions, which will reflect the type of self-testing, practice exercise and tutor marked assignment problems you are previously encountered. All areas of the course will be assessed. You ate to use the time between finishing the last unit and sitting for the examination to revise the whole course. You might find it useful to review your self-test, TMAs and comments on them before the examination. The end of course examination covers information from all parts of the course.

Course Marking Scheme

Assignment	**Marks**
Assignment 1-4	Four assignments, best three marks of the four count at 10% each- 30% of course marks
End of course examination	70% of overall course marks
Total	100% of course materials

Facilitator/Tutor and Tutorials

There are 16 hours of tutorials provided in support of the course. You will be notified of the dates, times and location of these tutorials as well as the name and phone number of your facilitator, as soon as you as you are allocated a tutorial group. Your facilitator will mark and comment on your assignments, keep a close watch on your progress and any difficulties you might face and provide assistance to you during the course. You are expected to mail you Tutor Marked Assignment to your facilitator before the schedule date. (at least two working days are required). They will be marked by your tutor and returned to you as soon as possible. Do not delay to contact your facilitator by telephone or e-mail if you need assistance The following might be the circumstances in which you would find assistance necessary, you would have to contact your facilitator if :

· Understand any part of the study or assigned reading
· You have difficulty with the self- tests
· You have a question or problem with an assignment or with the grading of an assignment You should endeavour to attend the tutorials. This is the only chance to have face to face contact with your course facilitator and to ask question which are answered instantly. You can raise any problem encountered in the course of your study. To gain much benefit from the course tutorials, prepare a question list before attending them. You will learn a lot from participating actively in the discussions.

Summary

Introduction to Expert Systems is a course that intends to provide concept of the discipline and is concerned with finding solutions through the software system on the basis of expert knowledge or provides an evaluation of known problems. Upon the completion of the course, you will be equipped with the knowledge of expert system and the application of its reasoning capabilities to reach a conclusion. You will be exposed to various application areas of expert systems. Furthermore, you will be able to answer the following types of questions:

· What is Expert System?
· What are roles of individuals who interact with expert system?
· What are various application areas of expert systems?

Of course a lot more question you will be able to answer.

I wish you success in the course and I hope you will find it both interesting and useful.

MODULE 1 Basic Concept of Expert Systems

Unit 1: What are Expert Systems?

1.0 Introduction

Formally, an expert system is defined as a knowledge-based computer program that exhibits, within a specific domain, a degree of expertise in problem solving that is comparable to that of a human expert. This problem method uses a knowledge base, which is carefully formulated on the basis of expert judgment, intuition, and experience.

2.0 Objectives

By the end of this unit, you should be able to :

- Define an expert system
- Identify the individuals who interact with expert system
- List the advantages and disadvantages of expert system
- State the features of expert system.

3.0 Definition of Expert System

Various definitions of expert systems have been offered by several authors:

- An expert system is software that attempts to provide an answer to a problem, or clarify uncertainties where normally one or more human experts would need to be consulted. ... en.wikipedia.org/wiki/Expert_system
- A type of artificially intelligent software system, which stores expertise concerning some subject matter in a knowledge base and attempts to answer questions or solve problems in a manner which simulates the thought processes of a human expert en.wiktionary.org/wiki/expert_system
- A type of application program that makes decisions or solves problems in a particular field by using knowledge and analytical rules defined by experts in the field. www.pcai.com/web/glossary/pcai_d_f_glossary.html
- A computer program that mimics the judgment of experts by following sets of chemical-specific knowledge rules derived from measured toxicity data and knowledge of how chemicals cause toxicity in animals and humans. www.epa.gov/opp00001/science/comptox-glossary.html
- A special program that resembles a collection of ``if ... then" rules. The rules usually represent knowledge contained by a domain expert (such as a physician adept at diagnosis) and can be used to simulate how a human expert would perform a task. mitpress.mit.edu/books/FLAOH/cbnhtml/glossary-E.html
- A computer program that uses knowledge and reasoning techniques to solve problems that normally require the abilities of human experts. Software that applies human-like reasoning involving rules and heuristics to solve a problem. **www.atlab.com/index.php/LIMS-Glossary-Terms-A-E.html**

- A system designed for solving problems in a particular application area. One can draw an inference from a stored knowledge base that was developed by recording and structuring human expertise through an individual commonly called a knowledge engineer. www.agriculture.purdue.edu/SSMC/Frames/newglossery.htm
- A software system with two basic components: a knowledge base and an inference engine. The system mimics an expert's reasoning process. **www.logisticsworld.com/logistics/glossary.asp**
- A domain specific knowledge base combined with an inference engine that processes knowledge encoded in the knowledge base to respond to a user's request for advice. www.expertise2go.com/webesie/tutorials/ESGloss.htm
- A computerized compilation of knowledge that is used to make "intelligent" decisions about the management or status of a process or system pdacrsp.oregonstate.edu/pubs/admin/admin_12/admin12.appA/admin12.appA.html
- An Expert System is a knowledge tool which uses rules and a process to guide the system user to specific information or knowledge which is indicated for the inquiry at hand. www.kwbsolutions.com/kbsterms.htm
- An artificial intelligence system that applies reasoning capabilities to reach a conclusion; a program that works with knowledge and relies on a database of if-then rules to draw inferences, in much the way a human expert does. **www.gregvogl.net/courses/mis1/glossary.htm**
- An Expert System (XPS) is a software system, which finds solutions on the basis of expert knowledge or provides an evaluation of known problems. Examples are systems for the support of medical diagnosis or for the analysis of scientific data. www.laendmarks.com/index.php
- Is a particular development of Artificial intelligence that helps to solve problems or make decisions through the use of a store of relevant Information (known as the Knowledge base, and derived from one or more human experts), and a set of reasoning techniques. www.markintell.com/language-business-intelligence/
- Artificial intelligence based system that converts the knowledge of an expert in a specific subject into a software code. This code can be merged with other such codes (based on the knowledge of other experts) and used for answering questions (queries) submitted through a computer.
- Formally, an expert system is defined as a knowledge-based computer program that exhibits, within a specific domain, a degree of expertise in problem solving that is comparable to that of a human expert. This problem method uses a knowledge base, which is carefully formulated on the basis of expert judgment, intuition, and experience. Thus an expert system embodies the cognition and ability of an expert in a certain realm, thus emulating the decision-making ability of a human

3.1 Roles of Individuals who interact with the Expert system

- **Domain expert :** The individuals who currently are experts in solving the problems; the system is intended to solve
- **Knowledge engineer** : The individual who encodes the expert's knowledge in a declarative form that can be used by the expert system;

- **User :** The individual who will be consulting with the system to get advice which would have been provided by the expert.
- System engineer - the individual who builds the user interface, designs the declarative format of the knowledge base, and implements the inference engine.

3.2 Expert System Features

The features which commonly exist in expert systems are:

■ Goal Driven Reasoning or Backward Chaining
An inference technique which uses IF-THEN rules to repetitively break a goal into smaller sub-goals which are easier to prove;

■ Coping with Uncertainty
The ability of the system to reason with rules and data which are not precisely known;

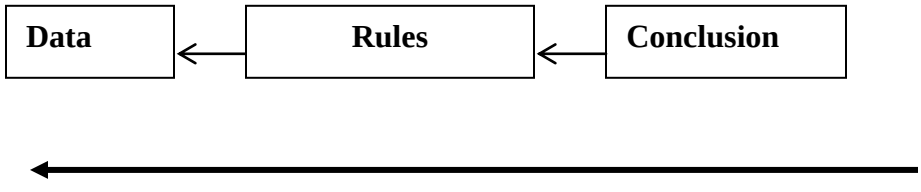
■ Data Driven Reasoning or Forward Chaining
An inference technique which uses IF-THEN rules to deduce a problem solution from initial data;

■ Data Representation
The way in which the problem specific data in the system is stored and accessed;

■ User Interface
That portion of the code which creates an easy to use system;

■ Explanations
The ability of the system to explain the reasoning process that it used to reach a recommendation.

■Goal-Driven Reasoning
Goal-driven reasoning, or backward chaining, is an efficient way to solve problems. The algorithm proceeds from the desired goal, adding new assertions found.



a = 1 if a = 1 & b = 2 then c = 3, if c = 3 then d = 4, d = 4 b = 2

Fig 1:
Source:

The knowledge is structured in rules which describe how each of the possibilities might be selected.

The rule breaks the problem into sub-problems.
Example :

KB contains Rule set :

Rule 1: If A and C
Then F

Rule 2: If A and E
Then G

Rule 3: If B
Then E

Then D
Rule 4: If G

Problem : prove

If A and B true Then D is true

■ Uncertainty

Often the Knowledge is imperfect which causes uncertainty.
To work in the real world, Expert systems must be able to deal with uncertainty.

- one simple way is to associate a numeric value with each piece of information in the system.
- the numeric value represents the certainty with which the information is known.

There are different ways in which these numbers can be defined, and how they are combined during the inference process.

■ Data Driven Reasoning

The data driven approach, or Forward chaining, uses rules similar to those used for backward chaining. However, the inference process is different. The system keeps track of the current state

of problem solution and looks for rules which will move that state closer to a final solution. The Algorithm proceeds from a given situation to a desired goal, adding new assertions found.

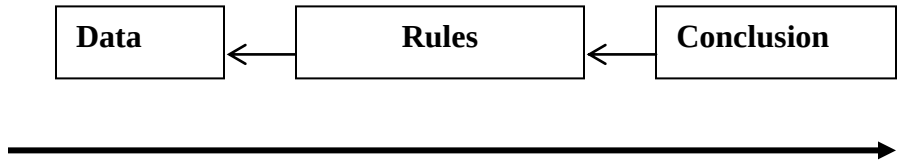


Fig 2
Source

The knowledge is structured in rules which describe how each of the possibilities might be selected. The rule breaks the problem into sub-problems.

Example :

KB contains Rule set :

Rule 1: If A and C
Then F

Rule 2: If A and E
Then G

Rule 3: If B
Then E

Rule 4: If G
Then D

Problem: prove

If A and B true Then D is true

■ Data Representation

Expert system is built around a knowledge base module.

- knowledge acquisition is transferring knowledge from human expert

to computer.
- Knowledge representation is faithful representation of what the expert knows.

No single knowledge representation system is optimal for all applications.

The success of expert system depends on choosing knowledge encoding scheme best for the kind of knowledge the system is based on.

The IF-THEN rules, Semantic networks, and Frames are the most commonly used representation schemes.

■User Interface

The acceptability of an expert system depends largely on the quality of the user interface.

- Scrolling dialog interface : It is easiest to implement and communicate with the user.
- Pop-up menus, windows, mice are more advanced interfaces and powerful tools for communicating with the user; they require graphics support.

■Explanations

An important feature of expert systems is their ability to explain themselves. Given that the system knows which rules were used during the inference process, the system can provide those rules to the user as means for explaining the results.

By looking at explanations, the knowledge engineer can see how the system is behaving, and how the rules and data are interacting.

This is very valuable diagnostic tool during development.

3.2 Benefits of Expert Systems

- Permanence: Expert systems do not forget.
- Reproducibility: Copies of an expert system can be made.
- Power: For applications where there is a maze of rules exhibited, it can be unraveled by the expert system.
- Efficiency: Expert systems can increase throughput and reduce personnel costs.
  -Expert systems are inexpensive to operate.
  -Development costs can be amortized over many years.
  - Expert systems can eliminate routine costs and reduce major maintenance costs.
- Consistency: With expert systems, similar events are handled the same way. Expert systems will make comparable recommendations for 'like' situations and are not affected by recent or primary effects.
- Documentation: Expert systems provide permanent documentation of the decision process.
- Completeness: An expert system can review all the transactions or possibilities.

- Timeliness: Fraud and/or errors can be prevented. Information is available sooner for decision making and action. The expert system works 24 hours a day, all year long.
- Scope: The expert system can encompass the cumulative expertise of many human experts.
- Business success: Owners reduce the inherent risks of conducting their business due to:
  - Consistency of decision making.
  - Documentation (ISO requirements)
  - Acquired expertise
- Positive impacts:
  - Productivity gains and cost savings.
  - Critical new tool for managers and a proactive answer to expertise attrition.
  - Decisions and solutions are more consistent and less subject to biases or sensitivity to the environment
  - Employment: shift to-wards more satisfying work.

3.2.1 Disadvantages of Expert Systems

- Common sense - In addition to a great deal of technical knowledge, human experts have commonsense. It is not yet known how to give expert systems common sense.
- Creativity - Human experts can respond creatively to unusual situations, expert systems cannot.
- Learning - Human experts automatically adapt to changing environments; expert systems must explicitly updated. Case-based reasoning and neural networks are methods that can incorporate learning.
- Sensory Experience - Human experts have available to them a wide range of sensory experience; expert systems are currently dependent on symbolic input.
- Degradation - Expert systems are not good at recognizing when no answer exists or when the problem is outside their area of expertise.

Activity A What is an Expert System?

4.0 Conclusion

This Unit has introduced you to the concept of expert system, its features and components. You have also been able identified the various individuals who interact with expert system.

5.0 Summary

In this Unit we have learnt that:

- An expert system is as a knowledge-based computer program that exhibits, within a specific domain, a degree of expertise in problem solving that is comparable to that of a human expert. This problem method uses a knowledge base, which is carefully formulated on the basis of expert judgment, intuition, and experience.

- **Domain expert :** is an individuals who currently is expert in solving the problems; the system is intended to solve
- **Knowledge engineer** : is individual who encodes the expert's knowledge in a declarative form that can be used by the expert system;
- **User :** is the individual who will be consulting with the system to get advice which would have been provided by the expert.

- System engineer – is the individual who builds the user interface, designs the declarative format of the knowledge base, and implements the inference engine

6.0 Tutor Marked Assignment

- Define an expert system
- Briefly state the role of individuals who interact with expert system
- List the advantages and disadvantages of expert system
- State the features of expert system.

7.0 Further Readings and Other Resources

Aikins JS, Kunz JC, Shortliffe EH, Fallat RJ., "PUFF: an expert system for interpretation of pulmonary function data.", Comput Biomed Res. 1983 Jun;16(3):199-208.

Barnett GO, Cimino JJ, Hupp JA, Hoffer EP. DXplain. An evolving diagnostic decision-support system. JAMA. 1987 Jul 3;258(1):67-74.

Basri. H, "An expert system for planning landfill restoration", Department of Civil and Structural Engineering, Universiti Kebangsaan Malaysia, Water Science and Technology Vol. 37, No. 8, pp 211–217, 1998.

Brusilovsky P. and Gorskaya-Belova T.B. (1992) The Environment for Physical Geography Teaching. Computers and Education, 1992, 18, (1-3), p.85-88.

Brusilovsky, P. (1996) Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6 (2-3), pp. 87-129. [PDF]

Brusilovsky, P. (2001) Adaptive hypermedia. *User Modeling and User Adapted Interaction*, Ten Year Anniversary Issue (Alfred Kobsa, ed.) **11** (1/2), 87-110

Unit 2 Components, History and the Development of Expert System

1.0 Introduction

This unit traces the history and the development of expert system. It discuses as well the component of expert system

2.0 Objectives

By the end of this unit, you will be able to :

- Trace the history of expert system
- Explain the various steps of developing an expert system
- Discus the components of expert system

3.0 Components of an Expert System:

Expert systems are usually built for specific application areas called Domains. The components of Expert System are as follows:

- User Interface.
- Inference Engine.
- Knowledge base.
- Working memory.
- Explanation facility

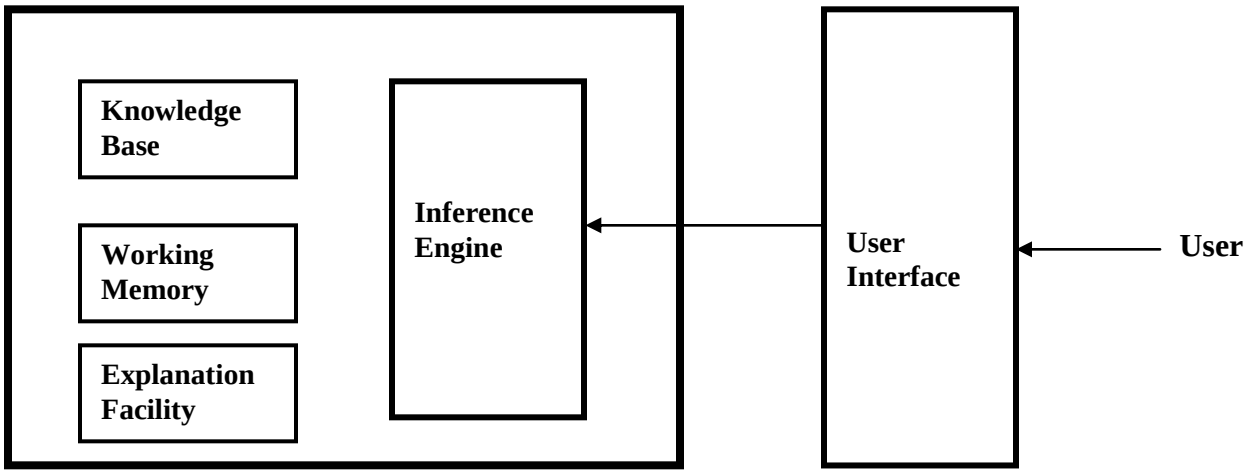


Fig 3 An Expert System:
Source:

3.0.1 The User Interface

- is the part of the system which takes in the user's query in a readable form and passes it to the inference engine. then displays the results to the user.

3.0.2 The Knowledge Base:

- is the collection of facts and rules which describe all the knowledge about the problem domain.
- derived from the human expert. Rules are typically structured as If/Then statements of the form:

IF <antecedent> THEN <consequent>

- Expert systems are also known as Knowledge-based systems.

3.0.2.1 Type of Knowledge

- shallow knowledge (experience) consists of all the peculiar heuristics and shortcuts

e.g.
IF it rains
THEN the vegetables will grow faster

- deep knowledge (theoretical) first principles, axioms, laws

e.g.
IF it rains
THEN the vegetables will grow faster
BECAUSE the soil will become more moist

3.0.3 Inference engine

- It is the brain of the expert systems that provides a methodology for reasoning about the information in the knowledge base, and for formulating conclusions.

- An inference engine tries to derive answers from a knowledge base(chooses which facts and rules to apply when trying to solve the user's query)

3.0.4 The working memory

- the working memory might be used to store intermediate conclusions and any other information inferred by the system from the data.

3.0.5 Explanation facility

- A part of the expert system that allows a user or decision maker to understand how the expert system arrived at certain conclusions or results

3.0.5.1 Explanation

- HOW

was a particular conclusion reached?

- WHY

the program asks the user a particular question?

- TRACE

displays all rules that are tried.

- WHAT-IF

explains what will happen if a certain value or rule is changed.

3.0.5.2 Inference Methods

- Chaining of IF-THEN rules to form a line of reasoning

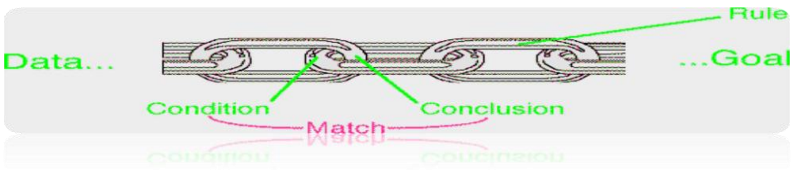


Fig 4
Source:

- There are two types of inferences

1- Forward chaining.
2- Backward chain*ing*

3.0.5.2.1 Forward Chaining

- The forward chaining inference engine takes rule, and if its conditions are true, adds its conclusion to working memory, until no more rules can be applied;
  - i.e.

if the conditions of the rule 'if A and B
then C' are true,
then C is added to working memory.

- In forward chaining the system simply test the rules in the order that occurs, therefore rule order is important.

3.0.5.2.2 Backward Chaining

- The backward chaining inference engines tries to prove a goal by establishing the truth of its conditions;
  - i.e.

the rule 'if A and B then C',
the backward chaining engine will try to prove C by first proving A and then proving B.

- Proving these conditions to be true, may well invoke further calls to the engine and so on.

3.1 History of expert system

In the mid-1960s, Edward A. Feigenbaum was one of the people in artificial intelligence research who decided, that it was expedient to know how much a computer program can know and that the best way to find out would be to try to construct an artificial expert. In the course of looking for an appropriate field of expertise, Feigenbaum across Joshua Lederberg, the Nobel laureate biochemist, who suggested that organic chemists sorely needed assistance in determining the molecular structure of chemical compounds.

In 1965, Lederberg and Feigenbaum together with Bruce Buchanan, started work on Dendral, the first expert system, at Stanford University. Conventional computer-based systems had failed to provide organic chemists with a tool for forecasting molecular structure. Human chemists know that the possible structure of any chemical compound depends on a number of basic rules about how different atoms can bond to one another. They also know a lot of facts about different atoms in known compounds. When they make or discover a previously unknown compound, they can gather evidence about the compound by analyzing the substance with a mass spectroscope-which provides a lot of data, but no clues to what it all means.

The only first major problem to be solved was Building the right kind of "if-then" program, with enough flexibility to use rules of thumb employed by human experts. After creating a program structure capable of manipulating expert knowledge, there is need to get some knowledge into the system. After feeding the computer program lots of data, the creators of Dendral interviewed as many expert chemists as they could to find out how they made their decisions. This "knowledge acquisition" phase has problems of its own. When asked how they know what they know, they're unable to articulate the answer. You just have to show them a program that makes decisions and ask them where the program is wrong, and why.

"Knowledge engineering" is the art, craft and science of observing human experts, building models of their expertise and refining the model until the human experts agree that it works. One of the first spinoffs from Dendral was Meta-Dendral, an expert system for those people whose expertise lies in building expert systems. By separating the inference engine from the body of factual knowledge, Buchanan was able to produce a tool for expert-systems builders.

In the mid-1970s MYCIN was developed by Edward H. Shortliffe, a physician and computer scientist at Stanford Medical School. The problems associated with diagnosing a certain class of brain infections was an appropriate area for expert system research and an area of particularly pressing human need because the first twenty-four to forty-eight hours are critical if the treatment of these illnesses is to succeed. With all its promise, and all its frightening ethical implications, medicine appears to be one of the most active areas of application for commercial knowledge engineering.

MYCIN's inference engine, known as E-MYCIN, was used by researchers at Stanford and Pacific Medical Center to produce Puff, an expert system that assists in diagnosing certain lung disorders. An even newer system, Caduceus, now has a knowledge base-larger than any doctor's-of raw data comprising about 80 percent of the world's medical literature.

Prospector, developed by SRI International, looks at geological data instead of molecules or symptoms. Recently this program accurately predicted the location of a molybdenum deposit that may be worth tens of millions of dollars.

1943	Post production rules: McCulloch and Pitts Neuron model
1954	Markov algorithm for controlling rule execution
1956	Dartmouth conference
1957	Perceptron by Rosenblatt, GPS by Newell, Shaw and Simon
1958	LISP by McCarthy
1962	Rosenblatt's principle of neurodynamics
1965	Resolution for automated theorem proving by Robinson Fuzzy logic by Zadeh
	DENDRAL (1st ES) designed by Feigenbaum and Buchanan
1968	Semantic nets and associative memory by Quillian
1969	MACSYMA: expert system in mathematics by Martin and Moses
1970	PROLOG by Colmerauer and Roussel
1971	HEARSAY I (speech recognition)
1973	MYCIN (medicine) by Shortliffe and al., followed by GUIDON (tutoring) by Clancey TEIRESIAS (explanation) by Davis EMYCIN (shell) by Van Melle, Shortliffe and Buchanan HEARSAY II (blackboard: multiple expert cooperation)
1975	Frames by Minsky
1976	AM (artificial Math.):creative discovery of math concepts by Lenat
	Dempster-Shafer theory of evidence for reasoning under uncertainty PROSPECTOR (mineral exploration) by Duda and Hart
1977	OPS expert system Shell by Forgy, used in XCON/R1
1978	XCON/R1 (DEC compzter config.) by McDermott
	Meta-Dendral (meta-rules and -induction) by Buchanan
	History (cont.)
1979	Rete algorithm for efficient pattern matching
	AI becomes commercial
	Inference Corp. Formed (releases ART expert system in 85)
1980	Symbolics (-> LISP machines)
1982	SMP math expert system Hopfield neural net Fifth generation project in Japan
1983	KEE expert system tool by Intellicorp
1985	CLIPS expert system tool by NASA

About two dozen corporations are currently selling expert systems and services. Teknowledge, founded by Feigenbaum and associates in 1981, was the first. IntelliGenetics is perhaps the most exotic, specializing in expert systems for the genetic engineering industry. Start-ups in this field tend toward science-fiction names-Machine Intelligence Corporation, Computer Thought Corp., Symbolics, etc. Other companies already established in non-AI areas have entered the field-among them, Xerox, DEC, IBM, Texas Instruments and Schlumberger.

Expert systems are now in commercial and research use in a number of fields:
• KAS (Knowledge Acquisition System) and Teiresias help knowledge engineers build expert systems.
• ONCOCIN assists physicians in managing complex drug regimens for treating cancer patients.
• Molgen helps molecular biologists in planning DNA experiments.
• Guidon is an education expert that teaches students by correcting answers to technical questions.
• Genesis assists scientists in planning cloning experiments.
• TATR is used by the Air Force in planning attacks on enemy air bases.

3.2 Development of an Expert System

The construction of an expert system is not as challenging as one might think, considering the enormous powers attributed to this class of programs. The task becomes easier because,

- Large portions of the Rule Translator can be generated automatically using lexical analyzer and parser generators, and
- Text editors (e.g., TextPad) can be purchased, inexpensively, and integrated into the Expert System Shell.

The design and the construction of the expert system involve the four major steps depicted in Figure below:

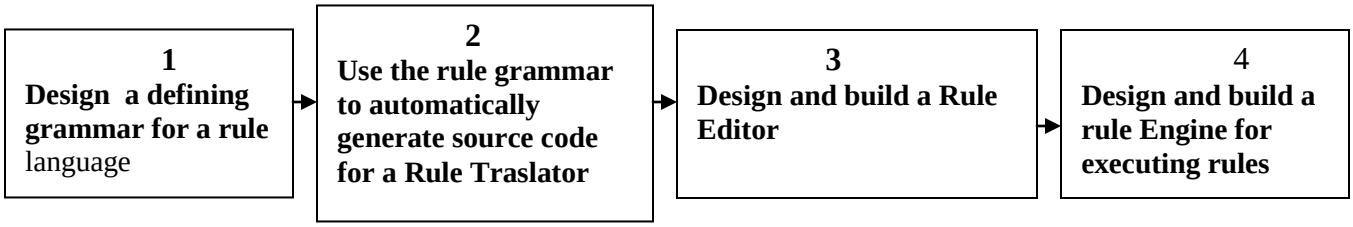


Fig 5: Construction Steps
Source:

3.2.1 Design a Rule Language

The best way to ascertain that the expert system is well-received is to design a rule language that is easy for users to learn and easy to use. The language must meet the needs of the subject matter expert. Subject matter experts need a language that mimics the way they speak, think, and operate. Its lexical elements and syntax rules must make the resulting language appear as natural (i.e., similar to the spoken language) as possible. These goals must be balanced with the need to create a language that can be recognized by translators created with conventional translation writing tools.

Rules must be expressed using a simple grammar, free of the idiosyncrasies of natural languages. Every sentence of the language used to express rules must have precise and unvarying meanings. A natural language with its inherent ambiguities is difficult to translate. This is because, in a natural language, words do not have a single meaning, nor are word meanings precise. In a natural language the meaning of words and phrases are colored by their context. Machine translation of natural languages is possible, but the program logic required is complex, expensive, and not wholly reliable. Machine translation of natural languages relies largely on heuristics that make translations imperfect.

On the other hand, context-free languages can be made to appear natural-like. However, as any programmer knows, these languages are unforgiving of spelling and grammar errors. Fortunately, methods for the translation of context-free languages are rooted in a mature and well-understood mathematics. Tools for building context-free languages translators make their construction straightforward and reliable. The tools used to generate translators for context-free languages require precise language definitions. These language definitions are called grammars.

The grammar defines both the syntax and the semantics of the language. The grammar defines the structure of a rule. This production is read: rule_definition produces the keyword, rule, followed by something called rule_name, followed by something called rule_description, followed by the keyword, is, followed by something called preamble_sequence, followed by something called statement_sequence, followed by the keyword, end, followed by something called rule_name.

3.2.2 Generate Translator from Rule Grammar

The rules composed by subject matter experts must be translated and stored in the Knowledge-base before they can be executed. Rules are transformed into machine-readable forms by the Rule Translator. The Rule Translator is generated automatically from lexical specifications and from the rule grammar. Lexical specifications define the lexical units (e.g., reserved words, numerical values, operators, etc) of the expert systems rule language. The rule grammar defines the syntax and semantics of the language.

Figure 10 illustrates the method by which the Rule Translator is automatically generated. The lexical analyzer generator depicted in this diagram operates on a definition of the languages lexical units to produce a software component referred to as a lexical analyzer or scanner. The scanner reads and converts rules in text form into a sequence of lexical units called tokens. Each token is a sequence of logically related characters having special meaning. For example, the text,

A stitch in time saves nine., might be decomposed by a lexical analyzer, designed for use by a grammarian or linguist, into the token sequence, article, noun, preposition, noun, verb, noun, period

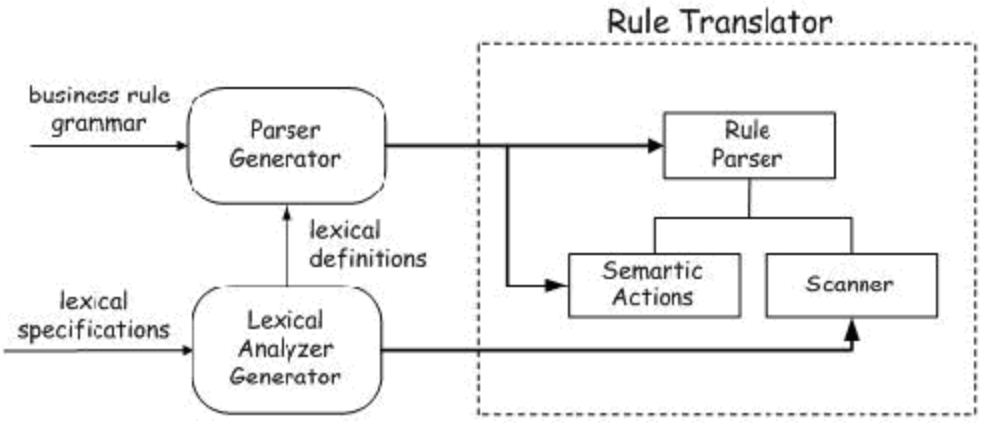


Fig 6
Source:

These token streams are used by the rule parser to structure the rule into larger, meaningful units called language constructs (e.g., control statements, assignment statements, etc.), and to perform the semantic actions (e.g., creation of AST elements) required by the constructs.

The parser generator produces two software components, a Rule Parser and a collection of semantic action subprograms. The Rule Parser, receives token streams from the Scanner. It structures these tokens into meaningful units (e.g., control statements, assignment statements, etc.). The parser invokes semantic actions (i.e., creation of AST elements) appropriate to the constructs it discovers. Semantic actions are software units that are dispatched by the parser as it

recognizes language constructs in the rule. These semantic actions build the Abstract Syntax Trees (ASTs) discussed in previous sections.

3.2.3 Design and Build a Rule Editor

The Expert Systems Rule Editor provides the means for subject matter experts to compose rules. Some people prefer graphical editors. Graphical editors with the expressive power needed to precisely specify complex rules, however, are difficult to build, and might very well end up being too unwieldy for subject matter experts to use. Text editors are simple to use and are rather easy to build. The disadvantage of text editors is this: they are unforgiving of spelling and syntax errors. This fact forces subject matter experts to understand rule syntax, simple though it might be. To relieve the subject matter experts of this burden and to speed development of rules, the editor could be equipped with special aids. These might include a syntax-directed editing feature, drop-down, menus, checkboxes, radio buttons, and help facilities.

Rather than build a text editor from scratch, commercially available editor similar to TextPad, could be purchased and substituted for a custom-built editor.

3.2.4 Design and Build a Rule Engine

The Rule Engine *executes* (or more properly, *interprets*) Knowledge-base rules. Execution of rules involves performing the steps diagrammed in Figure 11.

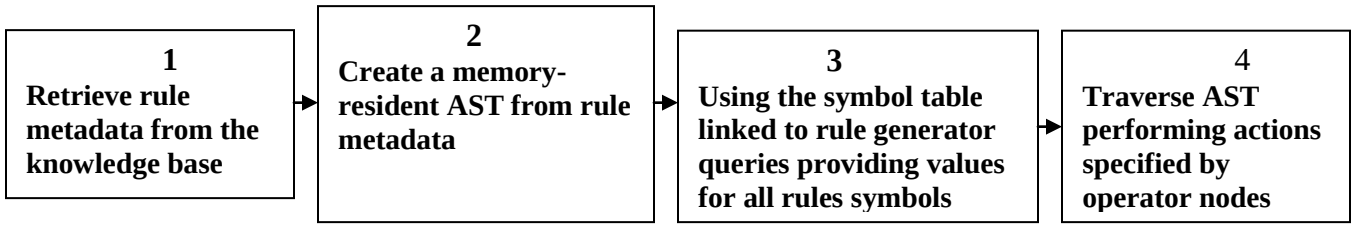


Fig 7: Rule execution steps
Source:

Rule *actions* may include:

- Assignment of values to an object (e.g., a row in a database table);
- Display of information (in reports or on screen) retrieved using ad-hoc queries specified in the rule; and
- Dispatch of built-in functions.

The retrieval of rule metadata and the construction of a memory-resident AST, operations corresponding to steps 1 and 2 depicted in Figure 11, could be performed by a single reference to a hypothetical method, Rule.AST, within the Rule Object Classes component:

rule_AST := Rule.AST(rule_identifier) ;

The method, RULE.AST, might query the Knowledge-base for elements of the rule identified by its input parameter, and assemble those elements into an attributed AST defining the rule.

To illustrate, consider the assignment statement,

E = x * (- (y + z)) ;

appearing in our sample rule. For the sake of simplicity, assume that the only purpose of the rule is to assign a value to the left-hand side of the assignment statement. Somewhere in that rule, attribute values for the variables, x, y, and z have been recorded.

The defining information for the terms x, y and z (as well as all other symbols referenced in the rule) is kept in a symbol table connected to the rule AST. If the information needed to assign values to x, y, and z is kept in a database, then database queries would be generated dynamically by the Rule Engine to obtain values for these variables. The information needed by the from clause and where clause of these queries is drawn from the rules symbol table. Suppose the queries return values of 200, 150, and 75 for x, y, and z, respectively.

The Rule Engine is now prepared to evaluate the rules AST, which in this case, looks something like the binary tree in Figure 12,

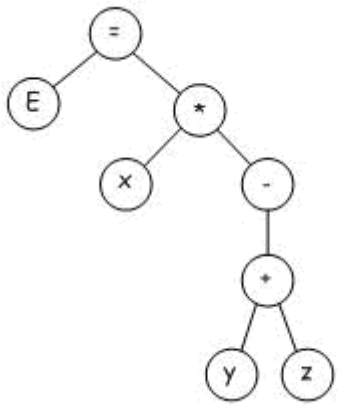


Fig 8: AST for statement, E = x *[- (y +z)]
Source:

The evaluation of the assignment operators right expression sub tree is performed using the method described by the algorithm shown in Figure 13.

```
Algorithm Evaluate_Expression_Subtree

  inputs: Arithmetic expression subtree,
          Values for each operand appearing in
          expression subtree
  outputs:  value of the arithmetic expression
            represented by the expression subtree
  notes:
begin
  traversal_stack_top ← Empty(traversal_stack) ;
  evaluation_stack_top ← Empty(traversal_stack) ;

  Traverse expression subtree in preorder, pushing subtree
    nodes to the "traversal" stack as they are visited ;

  while ( not Is_Empty(traversal_stack_top) ) do
    subtree_node ← Pop(traversal_stack) ;

    case ( Type(subtree_node) ) is

      when operand_node ⇒
        Push(subtree_node, evaluation_stack) ;

      when binary_operator_node ⇒
        p ← Pop(evaluation_stack) ;
        q ← Pop(evaluation_stack) ;
        result ← q <binary operator> p ;
        Push(result, evaluation_stack) ;

      when unary_operator_node ⇒
        p ← Pop(evaluation_stack) ;
        result ← <unary operator> p ;
        Push(result, evaluation_stack) ;

    end case ;
  end while ;
  expression_value ← Pop(evaluation_stack) ;

end Algorithm ;
```

Fig 9: Expression Evaluation Algorithm
Source:

on algorithm and attempts to show the machinations of this algorithm as it works to evaluate the arithmetic expression x*(-(y+z)).

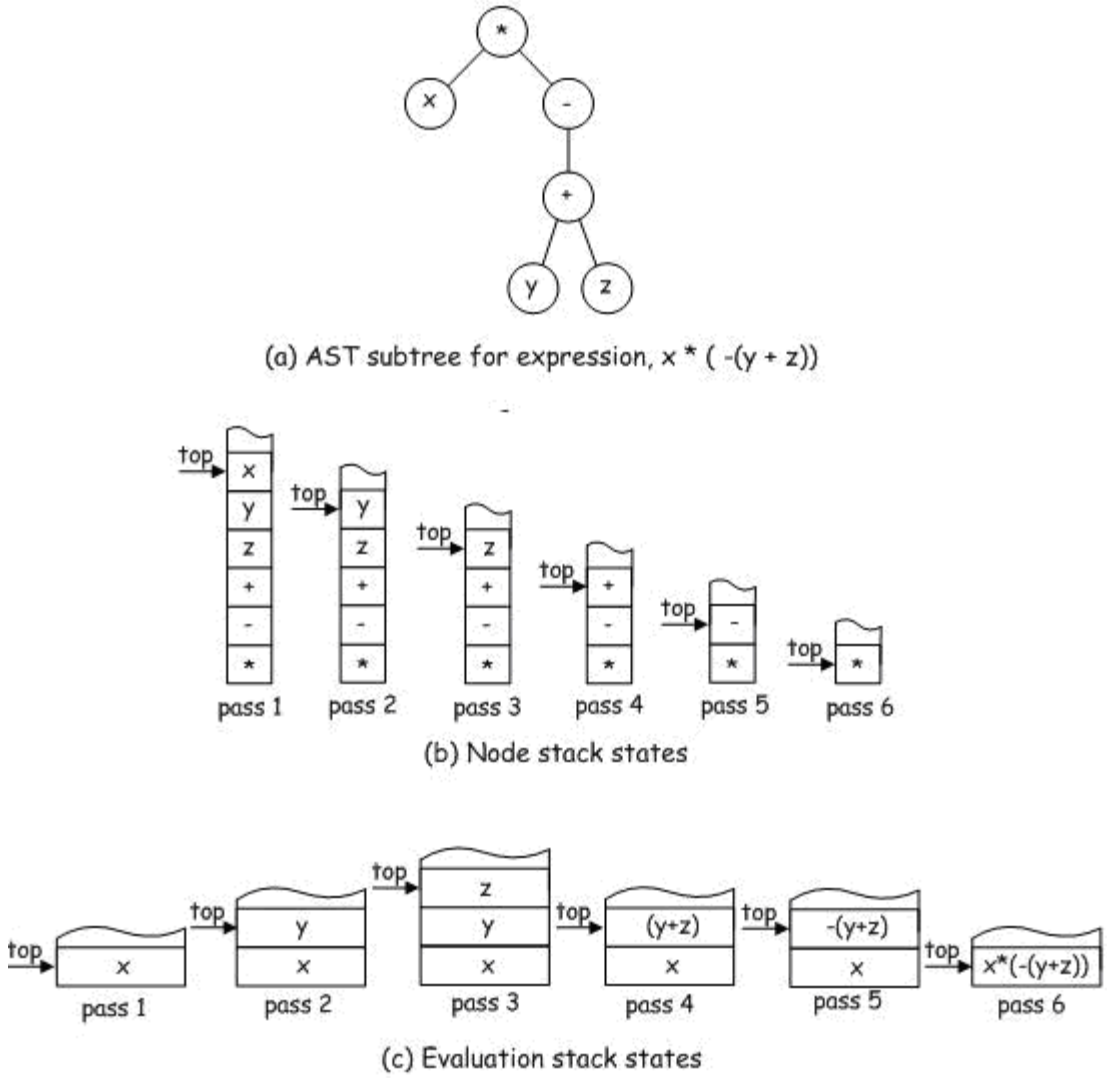


Fig 10 Trace of Expression evaluation algorithm execution
Source

Finally, the Rule Engine moves the value of the left-hand side expression to the destination variable, E. What it does afterward depends on how the rule was written.

The object of this last exercise was to eliminate any anxiety a developer might feel, by demonstrating the simplicity of the scheme by which rules are executed. Rules are not executed

by magic. We, simply, apply a variety of programming techniques that should be familiar to any trained computer scientist.

Activity B: `What is Inference Engine?

4.0 Conclusion

At the end of this unit, you have be able to:

- Trace the history of expert system
- Explain the various steps of developing an expert system
- Discus the components of expert system

6.0 Summary

- Expert system just like any other field has history which has been fully discussed in this unit. You have equally seen that components of expert system include user interface. inference engine, knowledge base., working memory, explanation facility

6.0 Tutor Marked Assignment

- Discus the history of expert system
- With the aid a well labeled diagram, discus the various step of developing an expert system.
- Discus the components of an expert system

7.0 Further Reading and Other Resources

Unit 3 Expert Systems Characteristics and Application

1.0 Introduction

These unit discuses the characteristics and the application of expert system

2.0 Objectives

By the end of this unit, you will be able to:

- Discus the characteristics of expert system
- Discus the application of expert system

3.0 Expert System characteristics

Expert system operates as an interactive system that responds to questions, asks for clarifications, makes recommendations and generally aids the decision-making process.
Expert systems have many Characteristics:

■ Operates as an interactive system

This means an expert system :

- Responds to questions
- Asks for clarifications
- Makes recommendations
- Aids the decision-making process.

■ Tools have ability to sift (filter) knowledge

- Storage and retrieval of knowledge
- Mechanisms to expand and update knowledge base on a continuing basis.

■ Make logical inferences based on knowledge stored

- Simple reasoning mechanisms is used
- Knowledge base must have means of exploiting the knowledge stored, else it is useless; e.g., learning all the words in a language, without knowing how to combine those words to form a meaningful sentence.

■ Ability to Explain Reasoning

- Remembers logical chain of reasoning; therefore user may ask
    1. for explanation of a recommendation
    2. factors considered in recommendation
- Enhances user confidence in recommendation and acceptance of expert system

■ Domain-Specific

- A particular system caters a narrow area of specialization; e.g., a medical expert system cannot be used to find faults in an electrical circuit.
- Quality of advice offered by an expert system is dependent on the amount of knowledge stored.

■ Capability to assign Confidence Values
- Can deliver quantitative information
- Can interpret qualitatively derived values
- Can address imprecise and incomplete data through assignment of confidence values.

■ Applications
- Best suited for those dealing with expert heuristics for solving problems.
- Not a suitable choice for those problems that can be solved using purely numerical techniques.

■ Cost-Effective alternative to Human Expert
- Expert systems have become increasingly popular because of their specialization, albeit in a narrow field.
- Encoding and storing the domain-specific knowledge is economic process due to small size.
- Specialists in many areas are rare and the cost of consulting them is high; an expert system of those areas can be useful and cost-effective alternative in the long run.

3.1 A Few Expert System Application Areas

3.1.1 Accounting & Finance - Cost Code Selector, Stock & Commodity Trading, Portfolio Construction, Home Purchasing, Financial Planner Training and Selection, Personal Tax Advisor, Detecting Insider Trading, Organizational Services, Credit Analysis Advisor, Bank Loan Identification, Credit Control, Loan Documentation, Assessment of Risk and Fraud in Financial Institutions, Aid in Tax Form Completion, Commercial Loan Approval Predictor...

3.1.2 Agricultural - Irrigation and Pest Control, Crop Variety Selection and Management, Soil Characterization and Utilization for Specific Areas, Fertilizer, Climate and Soil Interaction and Analysis, Salmon Stocking Rates and Species Selection, Forest Inventory, Weed Identification, Soil Conservation, Tree Selection Based on Environmental Conditions, Planning and Design of Agro forestry Systems...

3.1.3 Business - Alternatives for Fragmented Industry, Advertising Copy Development, Shipping Documentation and Routes, Market Advisor for Process Control Systems, Demographic and Market Assessment, Product Performance Trouble-shooting, Sales Personnel Assessment, Account Marketing, Invention Patent Ability, Salary & Benefit Planning, Client Profile Business Application Selection, Professional Service Selection, Career Goal Planning, Pension Fund Calculator, Unemployment Insurance Eligibility...

3.1.4 Chemical - Hazard Evaluation, Chemical Facilities Procedures, Correct Propellant Ingredient Mixture, Pollution Control Technology Permits, Common Metal and Alloy Identification, Real-time Process Controlled City Waste Water Management, Solvent Selection for Chemical Compounds, Pottery Glaze Recipe and Identification, Pulp Bleaching Advisor, Toxicity of Laboratory Chemicals, Lime Recognition System, Process Diagnosing and Troubleshooting...

3.1.5 Computer - Software System Diagnostic Modeling, Application Sizing, Software Quality Assurance, Program Classification, Locating Component Failure & Analysis, New Technology Selection, Training Systems, Custom Hardware Diagnostics, Decision Support Systems, MIS Support System, Program Library Maintenance, Fault Detection and Diagnostics of Wide Area Networks, Analysis of Statistical Data, Personal Computer Configuration, Shielding Technique Selection, Database Design, Computerized Technical Service Representation, Hardware and Software Selection by Non Technical Users, New User of Computer Assistance, Documentation Recommendations to Users, Monitoring, Repair Assistance and Problem Prediction of Operating System...

3.1.6 Construction - Pavement Rehabilitation & Design, Structural Damage Assessment, Equipment Evaluation & Selection, Material Costing & Selection, Project Scheduling, Cost Estimating, Evaluating Multifamily Housing Projects, Work Zone Safety Trainer/Advisor, Weld Procedure Selection and Cost Estimating, Soil Compacting, Fire Code Advisor, Alarm Management System...

3.1.7 Education - Training - Library Reference Material Recommendation, Interpretation of Statistical Quality Control Data, Teaching Mineral, Rock and Fossil Identification, Student Financial Aid Eligibility, Analysis of Metal Cautions, Fire Department Emergency Management Advisor, Medical Student Diagnostic Systems, Dentistry Advisor, Telephone Customer Support Instruction, Gas Turbine Training, Industrial Training, Patient Care Advisor for Student Nurses...

3.1.8 Engineering - Industrial - Engineering Change Control Demonstrating, Diesel Engine Lube Oil Wear Analysis, Superalloys Phase Analysis, Equipment Diagnostics, Electronic Semiconductor Failure Testing, Parts List Selection & Sizing, Power Generation System Scheduling, Component Failure Prediction, Machining Advisor, Numerically Controlled Machine Tool Selection, Petrochemical Plant Process Control, Control Panel Layout Design, Material and Process Design Advisor...

3.1.9 Insurance - Rating for Substandard Life Insurance, Workers Comp Classification, Underwriting Assistance, Social Security Help Desk and Benefit Identification, Unemployment Insurance Eligibility...

3.1.10 Medical - Admission Protocols, X-Ray Analysis, Hematological Diagnoses, Psychiatric Interviewing, Pediatric Auditory Brainstem Response Interpretation, Medical Decision Making, Respirator Selection for Preschool Children, Health Services Utilization and modeling, Diagnostic Systems, In-Vitro Fertilization, Symptom Analysis, Voice-Driven Lab Diagnosis, Rehabilitation Feasibility Strategies, Billing and Account Management, Disease Research...

3.1.11 Military, Government & Space Related - Submarine Approach Officer Training, Combat Methodology Selection, Radar Mode Workstation Designing, Federal Contract Management, Severe Weather Forecasting, Shuttle Payload On-Orbit Analysis, Metals Materials Selector, GB Satellite Abnormality Correction, Thermal Analyst, Selection of Non-materials in Aerospace Applications...

3.1.12 Trouble-Shooting - Airplane Starting Systems, Data Communications, Test and Repair of a PCB, Gas Turbine Control System, Bearing System Failures, Telecommunications Difficulties, Real-Time Process Control, Meterman's Assistant System, Mechanical Equipment and Systems Diagnosis, Weld Flaw Detection, Web Break Diagnosis in Paper Milling, Power Plant Turbine Generator Bearing Maintenance System...

3.1.13 Many, Many Others - Telephone System Configurator, Training Material & Product Selection, Manufacturing Resource Planning, Production Scheduling, Service Networking, Airline Scheduling, Cost/Benefit Analysis, Planning Implementation, Career Development, Quick Proposal Estimating, Credit Control, Product Development, Telephone Call Screening, Real Estate Market & Mortgage Credit Analysis, Retirement Planning, Sales Analysis...

The spectrum of applications of expert systems technology to industrial and commercial problems is so wide as to defy easy characterization. The applications find their way into most areas of knowledge work. They are as varied as helping salespersons sell modular factory-built homes to helping NASA plan the maintenance of a space shuttle in preparation for its next flight.

Activity C State the application areas of expert system

4.0 Conclusion

At the end of this unit, you have been able to learn:

- The various applications of expert system
- The characteristics of expert system

5.0 Summary

Expert system is found applicable to many arrears of human endeavours: Medicine, Agiculture,. Banking e.t.c. The characteristics of expert system include: operating as an interactive system tools have ability to sift (filter) knowledge, make logical inferences based on knowledge stored, possesses the ability to explain reasoning.

6.0 Tutor Marked Assignment

- Discus the various application of expert system
- Discus the characteristics of expert system.

7.0 Further Reading and Other Resources

Unit 4 Natural Language interface for Expert Systems

1.0 Introduction

In the last Unit, you have learnt about the characteristics and application of expert system. In this unit, you will learn about the natural language interface for expert system.

2.0 Objectives

By the end of this unit, you should able to:

- Define the natural language interface
- Explain the terms: speech recognition, speech synthesis, text-based pattern matching or gesture interaction as they relate to natural language interface.
- Discus the benefits of natural language interface in expert system
- Explain the shortcoming of natural language interface

3.0 Overview of Natural Language interface for Expert Systems

A Natural language Interface(NLI) is defined as an interface that has the ability to interact with users using human language such as English, as opposed to computer language, a command line interface, or a graphical user interface (Ogden and Bernick 1996; Zhou, Shaikh and Zhang 2004). The interface takes as input either written text or spoken speech. These types of input may be utilised individually or in a combination to produce a multimodal input interface. NLIs are usually only capable of understanding a restricted subset of a human language (usually restricted to a certain domain) and generate more or less pre-packaged responses (Patridge 1991). The user would have to learn to utilise a small subset of the English language in order to operate most of these interfaces (Cohen, Dalrymple, Tyler, Moran, Pereira, Sullivan and Gargan 1989; Androutsopoulos 2002). Some experts however feel that this is not an effective NLI as users would have to learn how to use the system before being able to utilise it effectively (Ogden *et al*. 1996). They argue that applications that utilise natural language should stimulate conversation between the human user and the computer in order to have a successful communication between the two parties. Human-to-human conversations take place by taking turns between speaker and listener. When NLIs are designed this should therefore be taken into account. NLIs were first utilised for human-computer interaction through natural language in 1966 with a system, called ELIZA, created by Joseph Weizenbaum. ELIZA was a simple conversational agent that was capable of parsing simple sentences and utilising them to pick out keywords. These keywords were then utilised for substitutions to turned into questions. ELIZA was not capable of holding a long conversation with a user as most statements made by the user were merely turned into questions. Though ELIZA did not utilise any sophisticated processing techniques, it was still a notable NLI as it was the first natural language interface.

Natural language can be processed by computers through *speech recognition*, *speech synthesis*, *text-based pattern matching* or *gesture interaction*. *Speech recognition* has come of age in recent years but has not matured to an extent that it can be utilised to have a conversation with a computer (Winograd and Flores 1986; Long 1994). Most speech recognition systems are restricted to certain keywords within a domain and, therefore cannot be utilised for accomplishing tasks outside a particular domain (Dusan and Flanagan 2004). Furthermore, these

systems do not adequately cover all utterances a user might utilise in order to accomplish a task in a certain domain. Research in the area of natural language processing is moving towards the creation of systems that are capable of learning human knowledge and obtaining related knowledge as a conversation proceeds (Dusan *et al*. 2004). *Speech synthesis* is the process of outputting simulated human speech (Dutoit 1997). Speech synthesis has been more successful than speech recognition and has a variety of applications. Contact centres use speech synthesis in the IVR to present the customer with menus with which they interact. Furthermore feedback is also provided to the customer using speech synthesis. *Text-based pattern matching* has mostly been utilised by conversational agents or search engines. These systems utilise pattern matching in order to interact with the user. This unit will introduce a conversational agent called ALICE, which utilises pattern matching techniques to converse with the user. Pattern matching is the act of checking text for a given structure and if it matches a certain task is performed. In the case of ALICE when a pattern is matched a template will be triggered. *Gesture interaction* is the process of analysing human gestures. Humans usually gesture when using human language in order to interact with other human beings. These gestures can convey the mood and sometimes the context of a conversation. In some cultures bowing in front of someone is a method of saying hello (therefore the context of the conversation at that point is introduction). Gesture interaction is usually utilised as a complimentary technique to speech (Cassell, Steedman, Pelachaud, Stone, Douville, Prevost and Achorn 1994). They provide a method through which context can be maintained in a conversation. Furthermore, the mood of the conversation can also be understood by the system.

3.1 Conversational Speech Interfaces

A conversational speech interface is one that utilises speech as an interaction technique and is an example of a natural language interface. This interface operates utilising the following functions: speech recognition (input) and speech synthesis (output). A major benefit of incorporating speech in applications is that it comes naturally to humans. Most people find speaking and listening easy. Though speech is easy for humans it is not so easy for computers. The reason for this is that speech technologies lack 100% accuracy (Lai and Yankelovich 2003), and there is a problem with ambiguity when utilising natural languages (Hendrix 1982). The main problem when designing these interfaces is that system developers do not design these systems with speech in mind from the beginning but rather in terms of the graphical user interface. This brings about a problem of merely designing a command line interface which utilises speech through a graphical user interface (Lai *et al*. 2003). Another crucial factor in determining whether or not the application will be successful is to determine whether there is a clear benefit to utilising speech (Lai *et al*. 2003). This involves assessing whether speech is absolutely necessary, in other words when the users hand or eyes are busy or when the task to be completed cannot be accomplished without the use of speech. Speech is not suitable in situations such as when large amounts of information need to be presented to the user.

3.1.1 Speech Recognition

Speech recognition can be seen as the process by which a computer can identify the parts of human speech (Jelinek 1997). The process starts with the user uttering something into a microphone and ends with the computer accomplishing a task. The solution that has not been successfully implemented by humans for computers is to accurately identify all possible words spoken by any person in any environment (Hill 1983; deWet, deVeth, Cranen and Boves 2003).

Systems performance in speech recognition can be affected by a number of factors including large vocabularies, multiple users, continuous speech and noisy environments. When a user speaks into a microphone, phonemes are extracted from the user speech. Phonemes are the linguistic units of human language (Zue, Cole and Ward 2000; Matthews 2002). These sounds are grouped together to form words utilised in human language. When the sounds are grouped together the actual process of understanding user input is initiated. Over the years, many approaches have been utilised but only two will be discussed here namely pattern matching and knowledge-based approaches. These two approaches are not mutually exclusive and are discussed below:

- Pattern matching - The goal of pattern matching is to take an unknown pattern and compare it to a set of known and stored patterns (also known as templates). These are established through training data. Templates are utilised to compare the pattern and compute a similarity score (Zue et al. 2000; Schroeder 2004). The template with the highest score is then chosen as it will have the highest acoustic similarity to the users input.
- Knowledge-based approaches - This approach makes use of a rule-based expert system which (as mentioned in Chapter 3) utilises a base classification of rules in order to function. However to utilise this model, a large set of rules would be needed in order to capture a great variability in speech (Kaminski 1989). Rules are formed from knowledge about speech signals. This approach could be useful, however, it does not perform effectively if there is insufficient knowledge.

3.1.2 Speech Synthesis

Speech synthesis enables computers and other electronic devices to output simulated human speech (Dutoit 1997). A computer system that is utilised for the purpose of producing human speech is known as a speech synthesiser. A text-to-speech (TTS) system is an example of a speech synthesiser that converts normal text into speech (Holmes and Holmes 2001). The quality and effectiveness of speech produced by these system are measured by utilising these characteristics (Karat, Vergo and Nahamoo 2003):

- Base-level achievement of speech that is intelligible (the ease with which the output is understood) to humans;
- produce speech that is as natural as that of human beings in other words how natural the speech is;
- produce speech that is personalised to a particular user, in other words it has the same intonation as a person‟s speech; and
- the final level and highest level of achievement is to produce speech based on a person‟s own voice recordings so that the speech sounds as if it belongs to that person.

For a text-to-speech system to function, a narrator is utilised to record a series of text (such as reading from an encyclopaedia, poetry, political news and various other texts). These narrations are carefully picked in order to ensure that every possible sound in a given language is recorded (Acapela Group 2008). These narrations are then sliced into the different phonemes found in the particular language and stored in a database. When the database is created, the various recorded utterances are segmented into one of the following: diphones (sound-to-sound transitions), syllables (units of organization for a sequence of speech sounds), morphemes (smallest linguistic

units that have semantic meaning), words, phrases and sentences (Dutoit 1997; Acapela Group 2008).

The above-mentioned process occurs in the *back end* of the TTS system. The *front end* has two major tasks namely: *normalization* and *text-to-phoneme conversion* (Santen, Sproat, Olive and Hirschberg 1997). *Normalization* is the process that occurs first, where the conversion of raw text occurs. All abbreviations and symbols are expanded into their respective words. These are then passed down to the component that starts the *text-to - phoneme conversion*. Once the text is passed on this component makes sure that the sentence is syntactically correct. The various phonemes are then extracted for that particular sentence and produced as human speech through a speaker or any other device capable of producing sound (Figure 4.1). There are two types of speech synthesis that are utilised for commercial applications namely concatenated synthesis and formant synthesis (Karat *et al*. 2003). Concatenated synthesis employs computers to assemble narrators recorded voices into speech signals. Though this sounds fairly simple, it is very database intensive and has large storage needs to store all recorded speech (Karat *et al*. 2003). Formant synthesis, on the other hand, utilises a rule-based expert system. This expert system applies a set of phonological rules to a specified audio waveform which simulates speech.

3.2 Text-Based Natural Language Interfaces

Text-based NLIs utilise text instead of speech in order to function. Text may be in the form of a query, sentence or a list of keywords (as used by search engines). A user will type in a question in an appropriate field (usually a textbox) and the system will retrieve information in accordance to the users query. Text-based natural language interfaces have been utilised in many applications including: databases (query and report generation) (Hendrix 1982; Wu, Ichikawa and Cercone 1996), conversational agents and search engines (matching user requests to keywords). Conversational agents are a communication technology that utilizes natural language and various linguistic methods to interact with human users through natural language (Lester *et al.* 2004). Conversational agents need to satisfy two sets of requirements for them to be effective (Lester *et al.* 2004):

- they must have good language processing capabilities such that they have the ability to engage in productive conversations with the user. This involves understanding user input and employing effective dialogue management techniques; and
- they must be scalable and reliable and allow for smooth integration within business processes.

Conversational agents are generally deployed on retail websites and are utilised by customers to enquire about products and services. However as these agents gained popularity they were deployed in a variety of other domains such as education (Ventura, Ventura and Olabe 2005), banking, travelling agencies and a variety of others (McBreen, Anderson and Jack 2000). Conversational agents have also been utilised in the virtual world and because these worlds are mostly text based in nature, they make an ideal environment for communication with the user. These types of virtual conversational agents can be found on social websites such as Secondlife (Lester *et al.* 2004). Another conversational agent known as ALICE utilises simple pattern-matching techniques to engage users in a conversation. ALICE utilises Artificial Mark-up Language (AIML), which is a derivative of XML to store its patterns and templates.

A Virtual World Personal Assistant Pattern matching is not the only technique utilised, various other methods such as Bayesian networks are also used. There are mainly two methods utilised by text-based interfaces to comprehend user input namely:

- *Pattern matching* – conversations and their responses are stored in pairs. The pattern is the user input or stimulus which is matched to produce a template which is the output of the system. The pattern can be seen as a simple text string that has to match the input exactly. The template is the output that a user receives as a response (as it was entered by the person who created the conversational agent). The biggest flaw with pattern matching is that it is difficult to maintain context of a conversation. Moreover it is impossible to cater for all possible inputs.
- *Bayesian networks* - is a graphical model that represents a set of variables and their probabilistic dependencies (Murphy 1998). It is implemented in conversational agents to maintain context in a conversation. Context can be maintained through Bayesian networks as the probability of the next question can be calculated. This probability is calculated by analysing current and previous user input. The question with the highest probability is usually always chosen.

3.3 Shortcomings of Natural Language Interfaces

Human-computer interaction experts believe that NLIs are not as attractive as they initially appeared to be (Long 1994). Early literature in this field focuses on the shortcomings of these interfaces in terms of user task completion (Fum, Guida and Tasso 1988; Cohen 1992). The shortcomings are brought upon by the ambiguous nature of a natural language and heavy dependence on a huge repository of knowledge (Fum *et al*. 1988). Consider the following quote: *At last, a computer that understands you like your mother* – 1985 McDonnell-Douglas ad. A computer can interpret this quote in a number of different ways (ambiguity) and thus will show us the difficulties in analyzing human language. If we look at the sentence carefully there are three interpretations that are possible (Lee 2004):

- the computer understands you as well as you mother understands you;
- the computer understands that you like your mother; and
- the computer understands you as well as it understands your mother.

This shows us how a simple sentence can be ambiguous for a computer, while we as humans can easily rule out all other alternatives except the first one. We do so, based on a great deal of background knowledge, including understanding what advertisements typically try to convince us of (Lee 2004). There are various techniques that could be used to get around this flaw. One of the oldest techniques used is to *limit the syntax and vocabulary* that exists in a natural language (Long 1994). Another technique that could be used is the *use of certain linguistic theories*, which outline certain rules that should be followed while conversing. Examples of such rules include:

- users should not give more information than necessary;
- users should not make unnecessary speech contributions than is necessary; and
- users should not intentionally make ambiguous references but rather use references that they believe will unambiguously describe exactly what they what want to achieve.

The most popular technique used to combat ambiguity is to *engage in some form of clarification or confirmation dialogue* to confirm if the interpretation is, in fact, the correct one (Cohen 1992).

3.4 Benefits of Natural Language Interfaces

Though NLIs have their disadvantages, they also have a variety of advantages. The main reason why NLIs are seen as beneficial is because no prior training is required in order to utilise them as people use natural language in order to interact with each other on a daily basis. Other benefits of conversational speech interfaces include (Hill 1971; Schroeder 2004; Zhou *et al.* 2004):

- Offers natural communication;
- Allows for physical mobility when troubleshooting problems that are not near the interface; and
- Allows for flattening of deep menu structures found in some computer systems. This would be possible as users could state their queries in natural language and would eliminate the need for menus. This would be beneficial for systems such as IVR.

3.5 Natural Language Interfaces for Rule-based Expert Systems

According to (Galina Datskovsky Moerdler et.al), two issues that must be considered when building a natural interface for an expert system are: semantic processing of the user statements and the design of an interpreter for the expert system that efficiently utilizes the facts entered by the user. The semantic approach is based on verb categorization and hierarchical structuring of each category. The parsing algorithm based on selectional restriction is directly encoded into each verb class hierarchy. The interpreter (Director) efficiently utilizes these facts for inferencing. Director uses a combination of forward and backward chaining that gives full consideration to each fact entered by the user and enables the system to process input in an efficient and focused manner.

With a natural language interface, a user may volunteer information at any point; either at the beginning of a session or as an answer to any question posed by the expert system. The expert system must be able to use this information to arrive at a solution quickly, avoiding inferencing and associated questions that are irrelevant to the user specified goal and information. Director has been designed specifically to support the construction of expert system with natural language interfaces. It allows for facts entered through natural language to be taken into account by using an efficient combination of forward and backward chaining. Forward chaining is used to ensure that full consideration is given to each fact asserted by the natural language module on behalf of the user. Backward chaining is controlled by heuristics that ensure a focused interaction. These heuristics are based on descriptions of how and when data has been entered and guide the selection of rules to be evaluated. Director also provides ready access to portions of its internal structure. This feature allows the system to answer certain types of queries with a minimal number of searches of the rule base. The interpreter is designed to come to a solution quickly, with the minimal number of questions posed to the user and is therefore

useful not only for systems with natural language interfaces, but also for systems whose problem domains involve data from hostile environments, such as nuclear reactors.

In order for a human expert to be able to answer a person's question he often has to carry out extensive dialogs with that person to gather information about his needs. Extensive interaction and clarification is also needed for expert systems. One way expert systems communicate with their users is via a menu interface [Shortliffe 76] [Clancey 79]. Unfortunately, these interfaces are often tedious or awkward [Datskovsky 84] and may even limit the capabilities of associated expert systems [Pollack et. al. 82].

A natural language interface can relieve some of the A typical rule-based expert system is constructed as a knowledge base and an associated interpreter (inference engine) [Hayes Roth 85]. The knowledge base is a collection of facts and production rules. A fact is a (name value) pair indicating the value of the object name; and a production rule is a statement of the form *IF premise THEN action.* Such a rule stores the knowledge that if the premise (left-hand-side) is true then the action (righthand-side) should be performed The interpreter controls the execution of the expert system by selecting and evaluating rules. As these rules are evaluated, they alter the values of the facts and generate input and output problems associated with the menu interface by allowing the user to receive advice in the most informative and least time consuming way. That is, overall session length should be shorter using natural language since the system will have to pose fewer questions. This is possible because natural language allows the user to volunteer more than just the requested information whenever the expert system presents a question to the user. The natural language module is responsible for interpreting incoming statements as facts. Although natural language interfaces to data base systems have been successfully constructed [Kaplan 79] [Woods et. al. 72] [Grosz ct al. 85], the task is more difficult in the expert systems domain. A semantic interpreter for a data base system usually relies on the regular structure of the data base as encoded in the schema describing it No such regularity or description is available in the expert systems' case. The lack of existing system structure makes the construction of domain independent semantics for expert systems difficult and means that some sort of structure that can be used as the basis for semantics must be built on top of the underlying rule base.

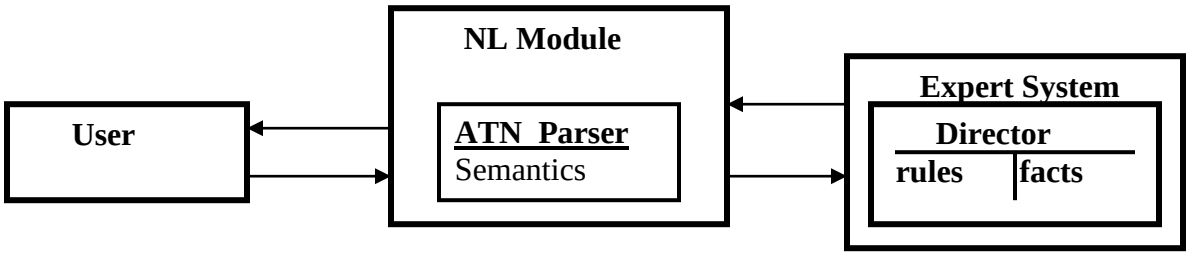


Fig 11: An Expert System illustrating a natural language interface
Source:

The semantics and control strategy under discussion resolves this problem. interpretations of natural language responses to system questions, deriving and making use of any additional facts volunteered by the user. Since question asking is normally determined by the sequence of rule firings in expert systems, a control strategy was developed and fully implemented in Director that determines which rules to fire so as to minimize the number of questions and to ensure that they are asked in a focused and coherent order.

Activity D. What is natural language interface?

4.0 Conclusion

This unit considers the concepts of natural language, the terms such speech synthesis, text-based pattern matching or gesture interaction as they relate to natural language interface. It considers the benefits and the shortcomings of natural interface.

5.0 Summary

In this unit, you have learnt that:

- A Natural language Interface(NLI) is an interface that has the ability to interact with users using human language such as English, as opposed to computer language, a command line interface, or a graphical user interface
- The shortcomings are brought upon by the ambiguous nature of a natural language and heavy dependence on a huge repository of knowledge .
- Natural language interface offers the following advantages:
    - Offers natural communication;
    - Allows for physical mobility when troubleshooting problems that are not near the interface; and
    - Allows for flattening of deep menu structures found in some computer systems. This would be possible as users could state their queries in natural language and would eliminate the need for menus. This would be beneficial for systems such as IVR.

6.0 Tutor Marked Assignment

- What is a natural language interface
- State the its advantages and shortcomings
- Discus the terms *recognition,* speech synthesis, text-based pattern matching or gesture interaction. and Speech recognition

7.0 Further Reading and Other Resources

Unit 5 Programming Language for Developing Expert System

1.0 Introduction

In this last unit, you learnt about natural language interface for expert system. You equally learnt about its benefits and shortcomings as well as various terms associated with natural language interface. In this unit, you will learn about Programming languages for developing expert system

2.0 Objectives

By the end of this unit, you will be able to:

- Identify various programming language tools for developing expert system
- State some of the distinguishing characteristics of programming languages needed for expert systems work.

3.0 Expert system shells

Most expert systems are developed via specialized software tools called shells. Shells provide framework to produce an expert system.so the knowledge base and rules are simply added to this frame work. These shells come equipped with an inference mechanism (backward chaining, forward chaining, or both), and require knowledge to be entered according to a specified format (all of which might lead some to categorize OPS5 as a shell). They typically come with a number of other features, such as tools for writing hypertext, for constructing friendly user interfaces, for manipulating lists, strings, and objects, and for interfacing with external programs and databases. These shells qualify as languages, although certainly with a narrower range of application than most programming languages. Expert system shells extensively in module 2

3.1 Special Programming Languages

Programming languages include conventional languages such as C++ and Java, as well as languages specifically designed for AI applications, these include LISP and PROLOG. There are two main classes of languages, procedural and declarative. Procedural languages include C++ and Java. They are relatively general purpose languages that can be used in many different programming situations, e.g Expert System, Microsoft Windows program and company specific applications. They are organized as a set of procedures, very similar to the way that a chapter in a book is divided into a series of paragraphs. While they offer no specific support for the development of expert system, they do offer the ultimate in flexibility. As such languages such as these can be used to develop systems where other tools may be inadequate. Expert systems are typically written in special programming languages. The use of languages like LISP and PROLOG in the development of an expert system simplifies the coding process. The major advantage of these languages, as compared to conventional programming languages, is the simplicity of the

addition, elimination, or substitution of new rules and memory management capabilities. Some of the distinguishing characteristics of programming languages

needed for expert systems work are:

• Efficient mix of integer and real variables
• Good memory-management procedures
• Extensive data-manipulation routines
• Incremental compilation
• Tagged memory architecture
• Optimization of the systems environment
• Efficient search procedures

Activity D What is declarative program?

4.0 Conclusion

This unit has introduced you to the various programming tools for building expert system.

5.0 Summary

In this unit, you have learnt that:

- The use of languages like LISP and PROLOG in the development of an expert system simplifies the coding process
- Shells provide framework to produce an expert system. so the knowledge base and rules are simply added to this frame work.

7.0 Tutor Marked Assignment

- What is expert System Shells
- Differentiate between procedural and declarative programme
- State the distinguishing characteristics of programming languages needed for expert systems work.

8.0 Further Readings and Other Resources

MODULE 2 Classes of Expert System

Unit 1 A rule-based expert system:

1.0 Introduction

A rule-based expert system is an expert system which works as a production system in which rules encode expert knowledge. Most expert systems are rule-based. It is an expert system based on a set of rules that a human expert would follow in diagnosing a problem. A classic example of a rule-based system is the domain-specific expert system that uses rules to make deductions or choices. For example, an expert system might help a doctor choose the correct diagnosis based on a cluster of symptoms, or select tactical moves to play a game.

2.0 Objectives

By the end of this unit, you should be able to:

- Define a rule based expert system
- Trace the history of Mycin
- Identify the components Mycin and trace its history
- Idendify the problem with Mycin
- State the practical use of Mycin

3.0 Mycin- a Rule Based Expert System

It is the name of a decision support system developed by Stanford University in the early- to mid-seventies, built to assist physicians in the diagnosis of infectious diseases. The system (also known as an "expert system") would ask a series of questions designed to emulate the thinking of an expert in the field of infectious disease (hence the "expert-"), and from the responses to these questions give a list of possible diagnoses, with probability, as well as recommend treatment (hence the "decision support-"). The name "MYCIN" actually comes from antibiotics, many of which have the suffix "-mycin".

3.2 History of Mycin

Mycin was originally developed by Edward Shortliffe for Stanford Medical School in the early- and mid-1970's. Written in Lisp, a language (a set of languages, actually) geared towards artificial intelligence, MYCIN was one of the pioneering expert systems, and was the first such system implemented for the medical field. The Goal of MYCIN was to compete in an experiment conducted at Stanford Medical similar to the Turing Test. The case histories of ten patients with different types of meningitis were submitted to MYCIN as well as to eight human physicians, including a resident, a research fellow, and five faculty specialists in infectious disease. Both MYCIN and the human physicians were given the same information. Both MYCIN's and the human physician's recommendations (as well as a record of the treatment actually received by the patients) were sent to eight non-Stanford specialists, completely unidentified as to which recommendation was MYCIN's and which were authored by the physicians. The outside

specialists gave MYCIN the highest score as far as accuracy of diagnosis and effectiveness of treatment.

The framework for MYCIN was derived from an earlier expert system called DENDRAL, created to find new chemical comounds in the field of orgainic chemistry (also developed at Stanford).

3.1 Components

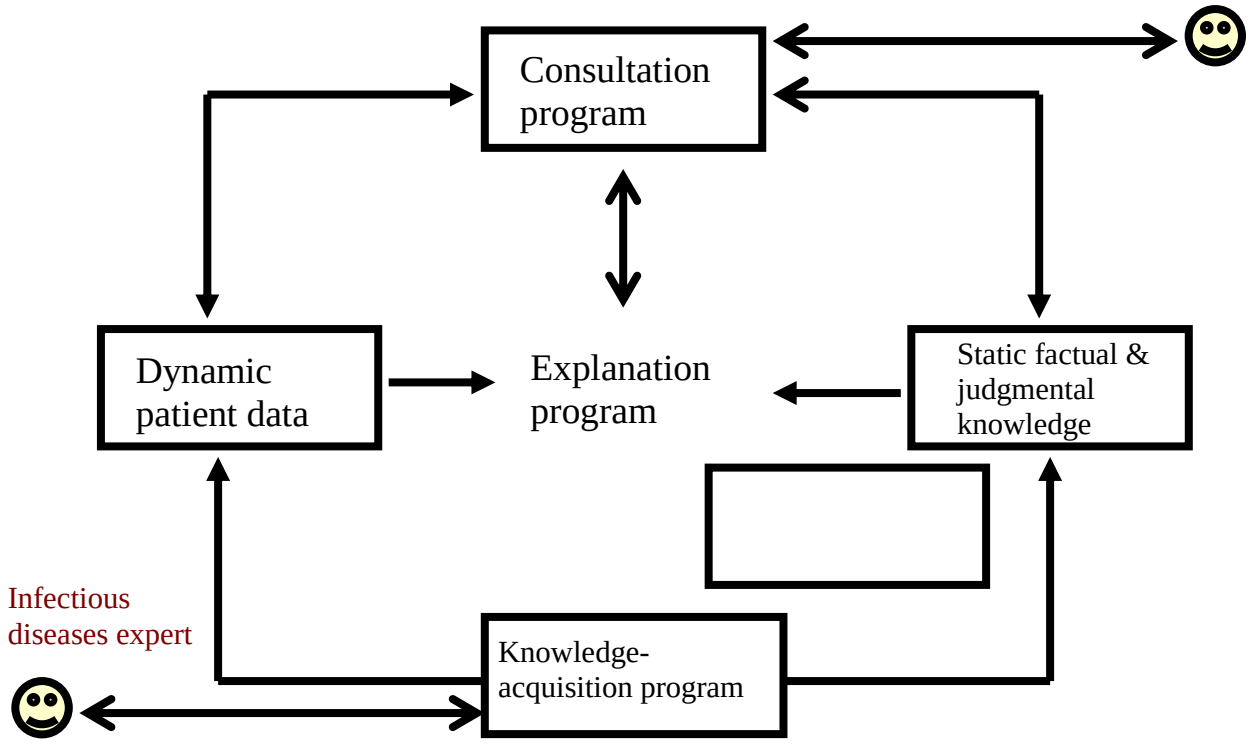


Fig 12: Component of Mycin System
Source:

The Knowledge Base

- Inferential knowledge stored in decision rules
  - If *Premise* then *Action* (*Certainty Factor* [*CF*])
  - If *A&B* then *C* (0.6)
  - The CF represents the inferential certainty
- Static knowledge:
  - Natural language dictionary
  - Lists (e.g., Sterile Sites)
  - Tables (e.g., gram stain, morphology, aerobicity)
- Dynamic knowledge stored in the context tree
  - Patient specific
  - Hierarchical structures: Patient, cultures, organisms
  - <*Object*, *Attribute*, *Value*> triples: <Org1, Identity, Strep>
  - A CF used for factual certainty <Org1, Identity, Staph, 0.6>

Mycin is an expert system comprised of two major components:

1. A knowledge base which stores the information the expert system "knows", much of which is derived from other information in the knowledge base.
2. An inference engine to derive knowledge from the presently known knowledge in the knowledge base.

Knowledge represented by MYCIN's knowledge base is represented as a set of IF-THEN rules with particualr certainty factors. For example:

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

Humans interface with MYCIN by answering a series of diagnostic questions akin to what a physician may ask a patient, as well as prompting for relevant test results. MYCIN takes this data as input and either arrives at a set of answers with respective probabilities, or branches to other questions in order to narro it's search. Researchers at Stanford found MYCIN to have an approximate correctness rate of 65%, which is better than the majority of physicians who are not specialists in diagnosing infections, and only slightly worse than physicians who were experts in that field (who had an average correctness of approximately 80%).

3.1.1 Mycin's knowledge base

It is small relative to those used by most rules-based systems today; it is on the order of ~500 rules. The science of the generation of these rules is known as "knowledge engineering". MYCIN uses a modification of the method of reasoning called "backward chaning" to search it's knowledge base. The modification comes in when the system is in the beginning stages of diagnosis, when the system asks a series of broad questions in order to weed out any unnecessary searches later on, such as checking for pregnancy if the patient is male. Only when the problem becomes more defined does MYCIN use full backwards chaining. If MYCIN checks a rule with a probability of 0.2 or less, it will abandon further searching on that particular rule.

3.1.2 The Inference Engine

The core of MYCIN, its inference engine, is called EMYCIN ("Essential MYCIN"). EMYCIN is the framework for MYCIN, a semi-separate system which could be used to create other rules-based expert systems to face problems similar to what MYCIN faces. In some cases this can be affected merely by changing the knowledge base.

3.2 Some Problems with MYCIN

If a computer can diagnose patients with a better rate of success than an average family doctor, why not implement it on a large scale? There are many problems associated with the implementation of an clinical expert system like MYCIN. The greatest of which is that it has never actually been used in a non-experimental way, i.e. it has never been actually used to diagnose and prescribe treatment for a patient. The reasons for this are twofold:

- If MYCIN gives an incorrect diagnosis and/or prescription (or, rather, a physician follows MYCIN's orders in giving such diagnosis and/or prescription), who would you sue?
- People tend to give too much credence to the recommendation of a computer simply based on the fact that it comes from a computer. This would of course lead some physicians to trust MYCIN too much in certain situations where they would be better served by taking it's recommendation with a grain of salt, so to speak.

The second list item here points to a broader range of problems governing expert systems, namely the ethics and philosophy of using them in environments where not only are they not used currently, but also where their decisions and recommendations have a substantial impact on human life.

Mycin was an expert system developed over 5 or six years in the early 1970s at the Stanford University; it was written in Lisp, by Edward Shortliffe under the direction of Bruce Buchanan and others; it derived from the earlier Dendral expert system, but considerably modified and extended the basic Dendral software. This computer system was designed to diagnose infectious blood diseases and recommend antibiotics, with the dosage adjusted for patient's body weight — the name derived from the antibiotics themselves, as many antibiotics have the suffix "-mycin".

Contents

3.3 Method

Mycin operated using a fairly simple inference engine, and a knowledge base of ~500 rules. It would query the physician running the program via a long series of simple yes/no or textual questions. At the end, it provided a list of possible culprit bacteria ranked from high to low based on the probablity of each diagnosis, its confidence in each diagnosis' probability, the reasoning behind each diagnosis (that is, Mycin would also list the questions and rules which led it to rank a diagnosis a particular way), and its recommended course of drug treatment.

Despite Mycin's success (see below), it is often now used as a cautionary tale in AI courses for people who generate their own ad hoc probability frameworks. Mycin had a limited depth to its reasoning hierarchy due to the noise introduced by its certainty factor system. This problem can be solved by using a rigorous probablistic framework, such as Bayesian statistics.

3.4 Results

Research conducted at the Stanford Medical School found MYCIN to have a correct diagnosis rate of about 65%, which was *better* than most physicians who were not specialists in diagnosing bacterial infections, and only slightly worse than those physicians who were themselves experts in the field (average correct diagnosis rate of about 80%).

3.5 Practical use

Mycin was never actually used in practice. This wasn't because of any weakness in its performance — in tests it outperformed members of the Stanford medical school. It was as much because of ethical and legal issues related to the use of computers in medicine — if it gives the wrong diagnosis, who can be held responsible? Issues with whether human experts would find it acceptable to use arose as well.

A difficulty that rose to prominence during the development of Mycin and subsequent complex expert systems has been the extraction of the necessary knowledge for the inference engine to use from the humans expert in the relevant fields into the rule base (the so-called knowledge engineering).

Mycin is a program that diagnoses infectious diseases. It reasons backward from its goal of determining the cause of a patient illness. It attempts to solve its goal of recommending a therapy for a particular patient by first finding the cause of the patient’s illness. It uses its production rule4s to reason backward from goals to clinical observations. To solve the top-level diagnostic goal, it looks for rules whose right sides suggest diseases. It then uses the left sides of those rules (the preconditions) to set up sub goals whose success would enable the rules to be invoked . these sub goals are again matched against rules, and their preconditions are used to set up additional sub goals.

Mycin is a well known rule based deduction system. Its expertise lies in the domain of bacterial Infections. Physicians usually must begin antibiotic treatment for patient who have bacterial infections without knowledge exactly which organism is the culprit. There is no time to wait for definitive laboratory culture evidence, which accumulates too slowly. For the desperately sick, therapy must begin at once – not 2 days from can either prescribe a broad – spectrum drug that covers all possibilities , or she can prescribed a better, disease – specific drug.

Mycin helps the physician to prescribe disease – specific drugs. Mycin in-forms it self about particular cases by requesting information from the physician about a patient's symptoms, general condition. History, and laboratory – test results that can be obtained easily and quickly. At each point, the question mycin asks is determined by Mycin's current hypothesis and the answers to all previous questions. Thus, the questions start as though taken from a checklist, but the questions then vary as evidence builds.

As we proceed through the processing stages of compute vision. We will no doubt be impressed by the similarities and parallel one can draw between vision processing and natural language processing. The - sensor stage in vision corresponds to speech recognization language understanding, the low and intermediate processing levels of vision correspond to syntactic and semantic language processing respectively, and high level processing, in both cases corresponds to the process of building and interpreting high level knowledge structures.

Activity E What is expert System shell?

4.0 Conclusion

This Unit has introduced you to rule based expert system, traced the history of Mycin. It has also discussed the components Mycin , the problem with it as well as its practical use.

5.0 Summary

In this unit, we have learnt that:

- A rule-based expert system is an expert system which works as a production system in which rules encode expert knowledge.
- Mycin is an expert system comprised of two major components:
    - A knowledge base which stores the information the expert system "knows", much of which is derived from other information in the knowledge base.
    - An inference engine to derive knowledge from the presently known knowledge in the knowledge base.
    - Mycin helps the physician to prescribe disease, its expertise lies in the domain of bacterial Infections.
    - Mycin major components are the Knowledge Base and the inference engine

6.0 Tutor Marked Assignment

- What is a rule based expert system
- Describe the components of Mycin
- State the practical use of Mycin
- Identify the problem with Mycin

7.0 Further Reading And Other Resources

Unit 2 Blackboard Expert System – HEARSAY

1.0 Introduction

The last unit exposed you to the concept of rule based expert system, its components using Mycin as an example. It also discussed the practical use of Mycin. In this unit, you will specifically learn about blackboard expert system with a particular emphasis on Hearsay

2.0 Objectives

By the end of this unit, you should be able to:

- Define a blackboard system
- Discus the components of blackboard system
- Identify the components of hearsay expert system
- Discus the benefit of blackboard architecture

3.0 Overview of Blackboard

A blackboard system is a task independent architecture for integrating multiple problem solving modules (referred to as knowledge sources). By task independent we mean that these architectures can be used for a wide range of tasks such as classification, design, diagnosis, repair etc. Integrated problem solvers, such as systems based on blackboard systems, attempt to overcome the inherent limitations of a single heuristic expert system by employing two or more problem solving (or reasoning systems). The problems solvers may all use the same technology or they may reply on different technologies. For example, a system might integrate a heuristic rule based reasoning system with a case-based reasoning system and possibly a model based system. However, integrated problem solving systems introduce a host of new issues. The overall problem facing such systems is that of integration of the problem solvers. Punch [1] categorises integration techniques as being fixed, programmable or task specific. However, this is possibly too simplistic a view of the integration problem. Integrated systems can possess different features in different areas, for example, one system might possess a flexible programmable structure, yet have a fixed cooperation strategy. Integration really refers to the problems of control, cooperation and communication of diverse problem solvers within a single system Blackboard systems (see [2]) are one (well established) approach to solving this
problem of control, cooperation and communication between problem solvers.

3.1 Methods

In a blackboard system, a set of problem solving modules (typically called knowledge sources) share a common global database (called the blackboard). The contents of the blackboard denote and indeed are often called hypotheses. These hypothesis are often structured hierarchically. Knowledge sources respond to changes on the blackboard, and interrogate and subsequently directly modify the blackboard. This modification results form the creation, modification and solution of hypotheses. In fact only knowledge sources are allowed to make changes to the

blackboard. It is therefore through the blackboard that the knowledge sources communicate and cooperate. In blackboard architecture, each knowledge source responds only to a certain class or classes of hypotheses. These hypotheses, that a knowledge source responds to, often reflect the different levels in the blackboard's hierarchy. The blackboard holds the state of the problem solution, while the knowledge sources make modifications to the blackboard when appropriate.

3.2 Component of Blackboard

A blackboard system consists of three components:

• Knowledge sources (KSs) are independent modules that contain the knowledge needed to solve the problem. KSs can be widely diverse in representation and inference techniques.
• The blackboard is a global database containing input data, partial solutions, and other data that are in various problem-solving states.
• A control component makes runtime decisions about the course of problem solving and the expenditure of problem-solving resources. The control component is separate from the individual KSs. In some blackboard systems, the control component itself is implemented using a blackboard approach (involving control KSs and blackboard areas devoted to control).

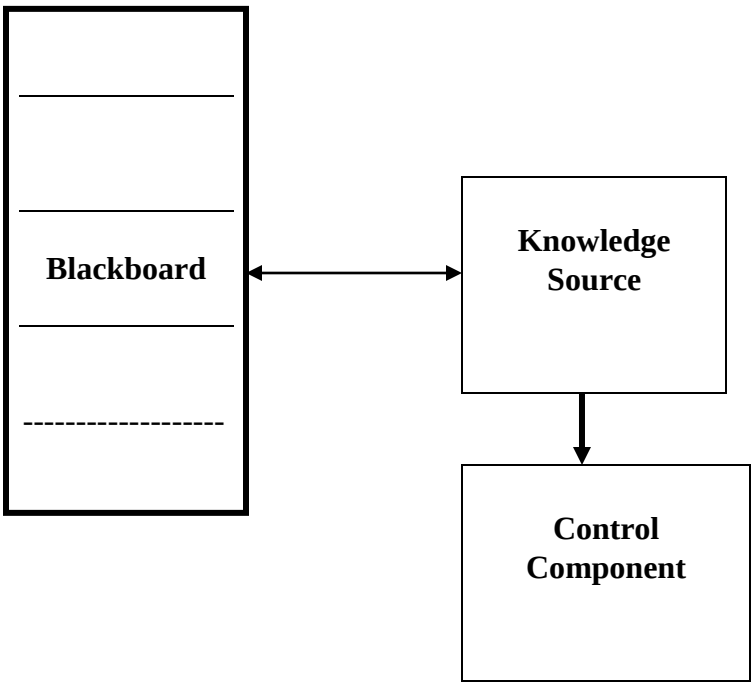


Fig 13: Basic Component of the Blackboard Model
Source:

3.3 Knowledge Source (KS)

Each KS is separate and independent of all other KSs. A KS needs no knowledge of the expertise, or even the existence, of the others; however, it must be able to understand the state of the problem-solving process and the representation of relevant information on the blackboard.

Each KS knows the conditions under which it can contribute to the solution and, at appropriate times, attempts to contribute information toward solving the problem. This knowledge that each KS has about when to contribute to the problem-solving process is known as a *triggering condition*.

KSs are much larger grained than the individual rules used by expert systems. While expert systems work by firing a rule in response to stimuli, a blackboard system works by firing an entire knowledge module, or KS, such as an expert system; a neural net or fuzzy logic routine; or a procedure.

Unlike our metaphor, KSs are not the active agents in a blackboard system. Instead, KS activations (sometimes called KS instances) are the active entities competing for execution resources. A KS activation is the combination of the KS knowledge and a specific triggering context. The distinction between KSs and KS activations is important in applications where numerous events trigger the same KS. In such cases, control decisions involve choosing among

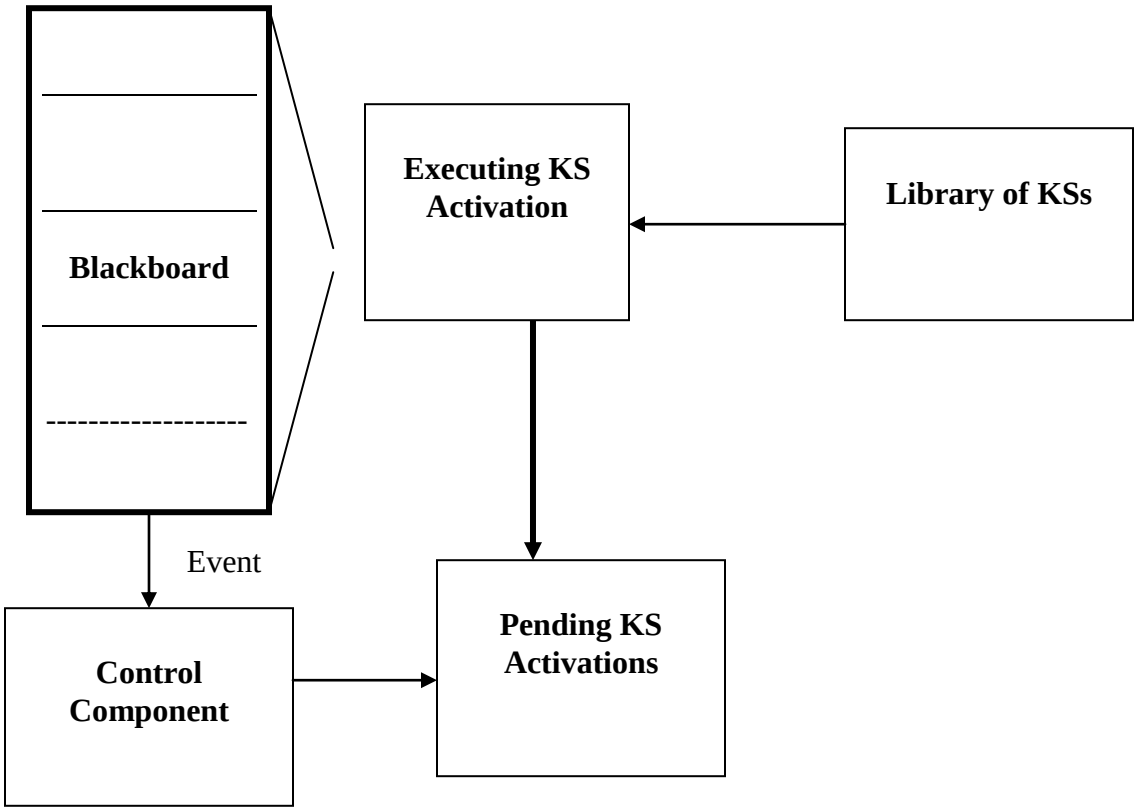


Fig 14: Basic Blackborad System
Source:

particular applications of the same KS knowledge (focusing on the appropriate data context), rather than among different KSs (focusing on the appropriate knowledge to apply). KSs are static repositories of knowledge, KS activations are the active processes.

3.4 The blackboard

The blackboard is a global structure that is available to all KSs and serves as:

- a community memory of raw input data; partial solutions, alternatives, and final solutions; and control information
- a communication medium and buffer
- a KS trigger mechanism.

Blackboard applications tend to have elaborate blackboard structures, with multiple levels of analysis or abstraction.
Occasionally, a system containing subsystems that communicate using a global database is incorrectly presented as a blackboard system. (A set of FORTRAN routines using COMMON is an extreme example of this view). True blackboard systems involve closely interacting KSs and a separate control mechanism.

3.5 Control component

An explicit control mechanism directs the problem-solving process by allowing KSs to respond opportunistically to changes on the blackboard database. On the basis of the state of the blackboard and the set of triggered KSs, the control mechanism chooses a course of action.
A blackboard system uses an incremental reasoning style: the solution to the problem is built one step at a time. At each step, the system can:

- execute any triggered KS
- • choose a different focus of attention, on the basis of the state of the solution.

Under a typical control approach, the currently executing KS activation generates events as it makes contributions to the blackboard. These events are maintained (and possibly ranked) until the executing KS activation is completed. At that point, the control components use the events to trigger and activate KSs. The KS activations are ranked, and the most appropriate KS activation is selected for execution. This cycle continues until the problem is solved.
Blackboard systems support a variety of control mechanisms and algorithms, so a choice of opportunistic control techniques is available to the application developer.

The blackboard approach has been applied in numerous areas, including the following:

- sensory interpretation
- design and layout
- process control
- planning and scheduling
- computer vision
- case-based reasoning

- knowledge-based simulation
- knowledge-based instruction
- command and control
- symbolic learning
- data fusion

In each of these applications, the scope of the problem to be solved was the prime factor in selecting a blackboard approach. That is, deciding whether to use a blackboard approach should be based on the problem-solving requirements discussed above, rather than the specific application area.

3.6 The Hearsay Expert System

Hearsay,1985, separates the different types of knowledge into coherent modules, called knowledge sources (KSs) in his speech signal processing system.The independent knowledge sources for phonetics, syntax, semantics were independent processes that looked at the speech signal and posted hypotheses about likely syllables, words, phrases, and so forth on the blackboard – a global data-structure accessed by all of the KSs through an established protocol. Hypotheses generated by the lower level knowledge sources (about syllables and words) would be examined for feasibility by the syntactic and semantic KSs; these, in turn, could make suggestions about what words might be expected and post them on the blackboard. The advantages of this organization are those generally associated with modularization of knowledge. The blackboard organization for representing multiple types of knowledge has been used in several domains besides speech understanding.

3.6.1 Hearsay-II

In the Hearsay-II blackboard system, the flow of control occurs between the knowledge sources. These knowledge sources attempt to work in an opportunistic manner, influenced by the scheduler. By opportunistic we mean that cooperation occurs dynamically determined during the execution of the system based on the current information available.

In a blackboard system, each of the knowledge sources monitors the blackboard looking for an opportunity to aid in the emerging solution being recorded on it. If a knowledge source identifies a situation in which it can provide some information it registers its wish to be activated by posting a knowledge source activation record to the scheduler's agenda.

The scheduler determines which of the knowledge sources wishing to be activated next is chosen for activation. It does this using information in the focus-of-control database and the likely impact of the knowledge source. The impact might be the degree to which it refines the hypothesis space. The blackboard monitor is the system component that updates the focus-of-control database. It does this by monitoring the generation and modification of hypotheses on the blackboard. Such a control regime can be described as flexible control.

3.6.2 Architecture of Hearsay-II

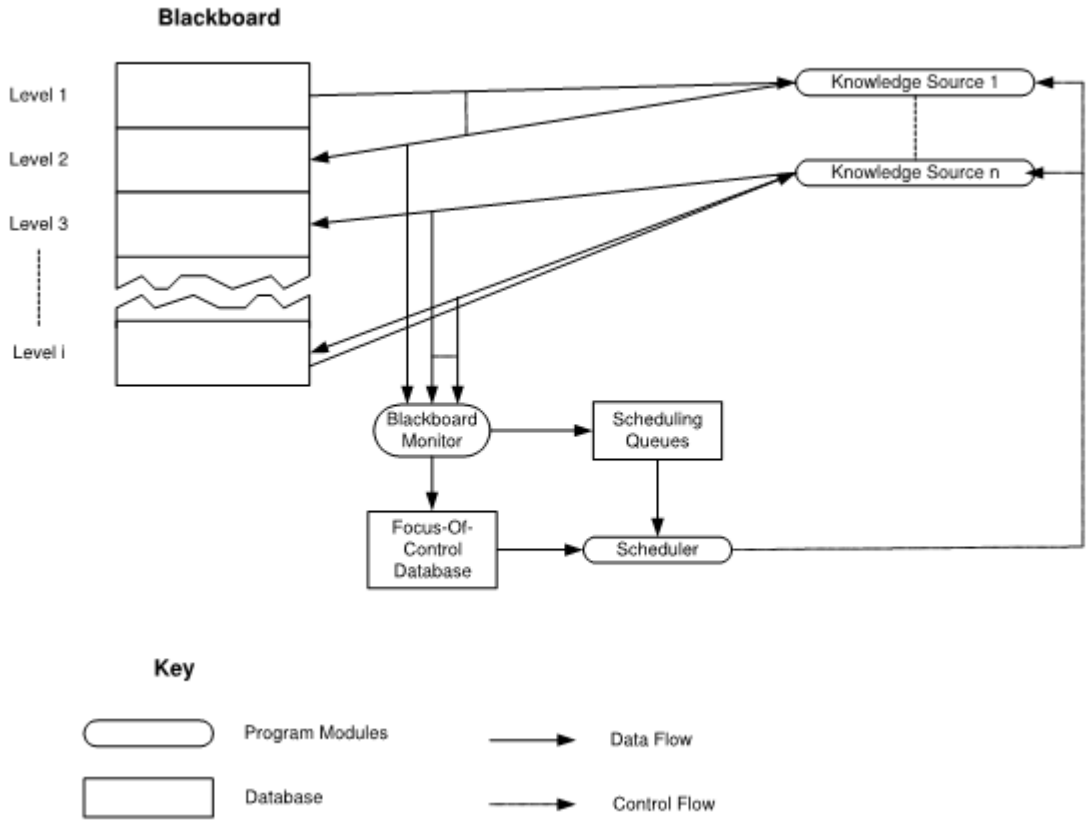


Fig 15: Architecture of Hearsay-II
Source: Erman et al

3.7 Benefits of blackboard architectures

Some of the main benefits of blackboard architectures and the systems built using a blackboard framework are as follows:

- **Flexibility of configuration.** The knowledge sources within a blackboard system are not tied together in a fixed or rigidly applied manner. Instead they communicate and cooperate via the common blackboard. This means that any knowledge sources can be added to the system without having to specify its existence in any other knowledge source.
- **Flexible problem solving.** A blackboard architecture is independent of any particular task and can therefore be used as the basis of very different applications, from route planning, to image processing, speech recognition and diagnosis. Cooperation is driven by the current state of the problem, rather then through any particular control strategy, and control of this cooperation is distributed between the knowledge sources themselves.

These knowledge sources then determine which they are appropriate and the scheduler determines which of the appropriate knowledge sources to actually activate.

- **Selection of knowledge sources**. Because more than one knowledge source may be able to perform the same function, the scheduler can select that problem solver which will provide the most benefit to the emerging solution. This can improve both problem solving efficiency and the quality of the eventual solution.
- **Multiple problem solvers.** The blackboard architecture promotes the integration and multiple problem solving modules (knowledge sources). Each of these knowledge sources can potentially have its own representation and inference mechanism. This means that the twin advantages of multiple reasoning systems are maintained, namely: resilient behaviour and efficient behaviour.
- **Management of multiple levels of abstraction**. The blackboard architecture explicitly supports the management of multiple levels of abstraction. Not only can the blackboard be divided hierarchically, but the knowledge sources can reflect this hierarchy. Problems of moving between the levels of the hierarchy can be overcome by providing knowledge sources whose sole task is to move information between two particular abstraction levels.
- **Opportunistic cooperation**. Cooperation ins a blackboard system is explicitly opportunistic; knowledge sources can post partial solutions to the blackboard in the hoped that some other knowledge source will be able to take these partial solutions and progress further down the path to the final solution.

3.8 Drawbacks of blackboard architectures

Some of the main drawbacks of blackboard architectures for integrating multiple problem solvers in a system such as a classification expert system include:

- **No communications language.** The blackboard acts as a very general communications facility. However, the communications language is the language of the hypotheses. That is, if all communications have to go via the blackboard then not only does the blackboard access mediate each decision step, it also constrains the representation of the information to be exchanged.That is, the language for passing information between problem solvers is the language of the intermediate hypotheses on the blackboard. This means that rather than an explicit request for some task to be performed, such as the generation of a simulation, the request must be phrased in the terms used to describe partial solutions on the blackboard – this may or may not be appropriate.
- **Computational complexity of cooperation.** The problem of determining which knowledge sources are appropriate in each situation and which of these appropriate knowledge sources should be activated has to be solved each time a decision has to be made. In the case of BB1 this can be computationally very expensive, even in Hearsay-II system this imposes a certain amount of computational overhead.
- **No guidance for task.** Blackboard architectures do not give any guidance to a system builder on how to solve a particular problem or perform a particular task within their framework. That is, blackboard architectures do not give any support for determining how specific problems should be addressed. Indeed they were never intended to do so, they are task independent architectures.
- **More complex problem solving framework.** The full facilities of a general problem

solving system are rarely required for a specific task. Yet blackboard architectures provide such a generality whatever the task being performed. Therefore, systems built within the blackboard framework often possess a more complex problem solving framework than is necessary.

- **No exploitation of task specific strategies.** There is no explicit support for task specific and domain specific problem solving strategies (for example, the use of useful diagnostic short cuts). Such strategies would have to be implicitly added into the scheduling system of the architecture.
- **Unbound nature of the control problem.** BB1 faces the potential unbound problem of determining which problem solver to invoke. A great deal of control "problem solving" could occur before any real work is begun on solving the problem on the domain blackboard.
- **Global database.** The blackboard, through which all knowledge sources communicate, by which cooperation is achieved and on which the emerging solution is stored, is actually a global database. It is directly modified by the knowledge sources and assumes that knowledge sources will behave responsibly. If they do not, chaos could ensue. This assumption goes against all the principles of information hiding which have been developed by software engineers to promote well engineered systems.
- **Knowledge of problem solving scope**. Although individual knowledge sources are aware of the types of situations (problems) they can address, no knowledge of the scope of the system as a whole is maintained. Therefore blackboard systems are unable to easily detect problems on or beyond the periphery of the whole system's ability.
- **Knowledge acquisition.** Although the use of specialist knowledge sources should make it easier to develop problem solving systems, and to encode the appropriate knowledge in them, no support is given to help in the process of obtaining this knowledge in the first place. In addition, each knowledge source can potentially possess its own knowledge representation that may require its own particular techniques not only to obtain but also to codify.
- **Implicit representation of management information.** Within the scheduler of the Hearsay-II system, problem management information is implicitly represented. This information focuses on the generation of a solution, and determines which knowledge source, from those capable of processing a particular solution, should be invoked next. In order to be able to do this efficiently, knowledge about problem solving strategies, short cuts, the abilities of the knowledge sources and task specific techniques are required. The implicit nature of the scheduler hides much of this information and some of these functions.
- **Problem solvers defined at system build time**. The knowledge sources need to be "connected" to the blackboard system at system build time. The abilities of the nowledge sources also need to be made available to the scheduler prior to the problem solving, in order that it can determine the "impact" of knowledge source.

Activity F What is a blackboard system?

4.0 Conclusion

In this unit, we have learnt about blackboard system, its architecture with a particular emphasis on hearsay expert system.

5.0 Summary

In this unit, we have learnt that:

- A blackboard system is a task independent architecture for integrating multiple problem solving modules (referred to as knowledge sources). By task independent we mean that these architectures can be used for a wide range of tasks such as classification, design, diagnosis, repair etc.
- A blackboard system consists of three components:
    1. Knowledge sources (KSs) are independent modules that contain the knowledge needed to solve the problem. KSs can be widely diverse in representation and inference techniques.
    2. The blackboard is a global database containing input data, partial solutions, and other data that are in various problem-solving states.
    3. A control component makes runtime decisions about the course of problem solving and the expenditure of problem-solving resources. The control component is separate from the individual KSs. In some blackboard systems, the control component itself is implemented using a blackboard approach (involving control KSs and blackboard areas devoted to control).
- Benefits of blackboard architectures include: Flexibility of configuration, Flexible problem solving, Selection of knowledge sources, Multiple problem solvers ,Management of multiple levels of abstraction, Opportunistic cooperation.

- Drawbacks of blackboard architectures include**:** Problem solvers defined at system build time, Implicit representation of management information, Knowledge acquisition, Knowledge of problem solving scope, No exploitation of task specific strategies, More complex problem solving framework, No guidance for task, Computational complexity of cooperation, No communications language

6.0 Tutor Marked Assignment

- What is a blackboard system?
- Discus the architecture of a blackboard system
- Discus the components of blackboard system
- Discus the components of hearsay expert system
- Explain the benefit of blackboard architecture

7.0 Further Reading and Other Resources

W. F. Punch, A Diagnosis System Using a Task Integrated Problem Solver Architecture (TIPS), Including Causal Reasoning, Ph.D. Thesis, The Ohio State University, 1989.

R. Englemore and T. Morgan, Blackboard Systems, Pub. Addison-Wesley Pub. Co. 1988.

L.D. Erman, F. Hayes-Roth, V.R. Lesser and D. R. Reddy, "The Hearsay-II speech understanding system: integrating knowledge to resolve uncertainty", ACM Computing Surveys 12(2), pp213-253, 1980.

L.D. Erman, P.E. London and S. F. Fickas, "The Design and an Example Use of Hearsay-

II", Proc. IJCAI-81, pp 409-415, 1981.
B. Hayes-Roth, "A Blackboard Architecture for Control", Artificial Intelligence 26, pp 251-321, 1985.

Unit 3 Frame Based Expert System- KEE

1.0 Introduction

The last unit exposed you to blackboard system with a particular emphasis on Hearsay. In this Unit, you will learn about Frame based expert system as well as various terms associated with it.

2.0 Objectives

By the end of this unit, you should be able to:

- Define the Frame based expert System
- Frame, Slots, Attribute, Database, Class-Frame, Instance and Slots associated with frame.
- Discus KEE and its components

3.0 Overview of Frame based Expert System

A frame is a data structure with typical knowledge about a particular object or concept. Frames were first proposed by **Marvin Minsky** in the 1970s. Each frame has its own name and a set of **attributes** associated with it. *Name*, *weight*, *height* and *age* are slots in the frame *Person*. *Model*, *processor*, *memory* and *price* are slots in the frame *Computer*. Each attribute or slot has a value attached to it.Frames provide a natural way for the structured and concise representation of knowledge. A frame provides a means of organising knowledge in **slots** to describe various attributes and characteristics of the object.Frames are an application of **object-oriented programming** for expert systems. The concept of a frame is defined by a collection of **slots**. Each slot describes a particular attribute or operation of the frame. Slots are used to store values. A slot may contain a default value or a pointer to another frame, a set of rules or procedure by which the slot value is obtained. Frames includes information about how to use the frame, information about expectations (which may turn out to be wrong), and information about what to do if expectations are not confirmed, and so on. Collections of such frames are to be organized in frame systems in which the frames are interconnected. Frame-based systems are knowledge representation systems that use frames, as their primary means to represent domain knowledge. Knowledge is organized in the form of chunks called frames. These frames are supposed to capture the essence of concepts or stereotypical situations. For example, being in a living room or going out for dinner, by clustering all relevant information for these situations together.

Frame Based Expert System is associated with the objects of interest and reasoning consists of confirming expectations for slot values. Such systems often include rules too. In a frame-based system, the inference engine also searches for the goal, or inother terms for the specified attribute, until its value is obtained.In a rule-based expert system, the goal is defined for the rule base. In a frame-based system, rules play an auxiliary role. Frames represent here a major source of knowledge, and both methods and demons are used to add actions to the frames. Thus, we might expect that the goal in a frame-based system can be established either in a method or in a demon.

- A demon has an IF-THEN structure. It is executed whenever an attribute in the demon's IF statement changes its value. In this sense, demons and methods are very similar, and the two terms are often used as synonyms.
- However, methods are more appropriate if we need to write complex procedures. Demons, on the other hand, are usually limited to IF-THEN statements.

3.1 Terms Associated with Frame Base Expert System

3.1.1 Frame: A data structure with typical knowledge about a particular object.

3.1.2 Slot: A component of a frame in a frame-based system that describes a particular attribute of the frame.

3.1.3 **Attribute:** A datum associated with an entity.

3..1.4 Object Oriented Programming

3.1.4.1. Class: A composite type which can contain both data and subprograms, and which can participate in inheritance.

3.1.4.2 object-oriented programming: A style of programming that defines data as objects with attributes and methods that are applied to those objects, and which can be inherited by other objects.

3.1.4.3 Attribute: A datum associated with an entity.

3.1.5 Database

- **Table:** A set of data elements that has a horizontal dimension (rows) and a vertical dimension (columns) in a relational database system.
- **Row:** A group of fields or columns related by a common identifier.

Each row of a database table contains information about an entity.

3.2 Frames As A Knowledge Representation Technique

3.2.1 Frames

3.2.2 Class-Frame: A frame that represents a class.
3.2.3 Instance-Frame: A frame that represents a instance.
3.2.4 Facet: A component of a slot in a frame.

3.3 The Slots Of A Frame Can Include:

- Frame name
- Frame hierarchy: parent frame(s)
- An attribute value. This can be a number, a boolean value, or symbolic (character).
- A default value for the attribute.
- The range of valid values for the attribute.

- A subprogram which is executed when the slot value is changed or accessed. These subprograms are called demons in the ai world. (These are not the same as the demons, or daemons, of the UNIX world).

3.4 A demon can be triggered when:

- A new value is added to a slot.
- A value is removed from a slot.
- The value in a slot is replaced by a new value.
- A value is requested from an empty slot (which does not contain a value).

3.5 Demons are facets:

The term "frame" is used to refer both to class-frames, which are abstract templates, and to instance frames, which are concrete analogs of a particular entity.

3.6 Object Oriented Programming

- **Class:**A composite type which can contain both data and subprograms, and which can participate in inheritance.
- **Object:**A variable which is an instance of a class.
- **Member:**A data value (attribute) or function (method) contained in a class or structure.

- The class of OOP corresponds to the class-frame of frames.
- The object of OOP corresponds to the instance-frame of frames.
- There is no OOP term which refers both to classes and objects.
- The concept of a frame slot is similar to the concept of an OOP class member, but is slightly broader.
- The frame name is considered a frame slot, but the class name is not considered a member of a class or object.
- The frame hierarchy is considered a frame slot, but the class hierarchy is not considered a member of a class or object.
- Preferred OOP style always accesses data members through methods (functions), which can be used to supply default values and perform range checks.
- The Common Lisp Object System (CLOS) of Common Lisp also uses the term slot to refer to parts of an object or class.

3.7 Database

- Triggers (procedures) can be attached to tables. These are executed when new rows are inserted into the table, when a row is deleted, or when a row is updated (changed.)
- Databases can specify default values
- Item`Some databases can specify custom data domains which can be used to provide range checking.

3.8 Inheritance in Frame-Based Experts

3.8.1 Frames

In the context of frames, inheritance refers to:

- The inheritance by a class-frame of the characteristics (slots) of all its superclasses.
- The inheritance by an instance-frame of the characteristics of its class-frame.

3.8.2 Object Oriented Programming

Inheritance: The implicit inclusion of members of a parent class in a child class.
Multiple Inheritance: The inheritance of a class from more than one parent class.
Single Inheritance: The inheritance of a class from only one parent class.
Subclass A class which is derived from another class.
Superclass:A class from which the current class is derived.
Parent:A direct superclass

3.8.2.1 In the context of object oriented programming, inheritance refers to:

- The inheritance by a class of the members of all its superclasses.
- When a class has a single parent, which may itself have a single parent, and so on, it is a case of single inheritance, even though the class may have many superclasses.
- Multiple inheritance means more than one parent, or direct superclasses.
- Multiple inheritance is problematic.
- If different parent class have the same member, ambiguities or duplication can occur.
- C++ allows multiple inheritance, but the usage is complex, and care should be used.
- Java does not allow multiple inheritance of classes. It does allow multiple interitance from interfaces, which have no data members and only abstract (undefined) methods.)
- The Common Lisp Object System (CLOS) of Common Lisp does support multiple inheritance.

3.8.3 Database:A database can contain views, or virtual tables, which combine data from two or more tables.

3.8.4 Summary

This table shows approximate, conceptual **correspondence between terms in the AI, OOP, and Database contexts:**

AI Frames	OOP	Database
class frame	class	table
instance frame	object	row
slot	member	
attribute	field	column
demon	method	trigger

3.9 Overview of KEE`

KEE (Knowledge Engineering Environment) is a frame-based development tool for Expert Systems[1]. KEE was developed and sold by IntelliCorp. It was first released in 1983 and ran on Lisp Machines. KEE was later ported to Lucid Common Lisp with CLX (X11 interface for Common Lisp). This version was available on various Workstations.

On top of KEE several extensions were offered:

- **Simkit,** a frame-based simulation library
- **KEEconnection**, database connection between the frame system and relational databases.

Frames are called *Units* in KEE. Units are used for both individual instances and classes. Frames have *slots* and slots have *facets*. Facets for example describe the expected values of a slot, the inheritance rule for the slot or the value of a slot. Slots can have multiple values. Behavior can be implemented using the message-passing paradigm. KEE provides an extensive graphical user interface to create, browse and manipulate frames. KEE also includes a frame-based rule system. Rules themselves are frames in the KEE knowledge base. Both forward and backward chaining inference is available.

KEE supports non-monotonic reasoning through the concepts of *worlds*. Worlds allow provide alternative slot-values of frames. Through an assumption-based Truth maintenance system inconsistencies can be detected and analyzed.[5]

ActiveImages allows graphical displays to be attached to slots of Units. Typical examples are buttons, dials, graphs and histograms. The graphics are also implemented as Units via ***KEEPictures*** - a frame-based graphics library.

3.9.1 Components of KEE system

KEE Knowledge Engineering Environment. Frame-based expert system. Supports dynamic inheritance, multiple inheritance, polymorphism. Classes, meta-classes and objects are all treated alike. A class is an instance of a meta-class. Can control rules for merging of each field when multiple inheritance takes place. Methods are written in LISP. Actions may be triggered when fields are accessed or modified. Extensive GUI integrates with objects. Can easily make object updates to be reflected on display or display selections to update fields. This can in turn trigger other methods or inference rules which may then update other parts of the display. Intellicorp, for TI Explorer. "The Role of Frame-Based Representation in Reasoning", R. Fikes et al, CACM 28(9):904- 920 (Sept 1985).

Activity G: What is KEE?

4.0 Conclusion

This unit has introduced you to Frame based expert system as well as various terms associated with it.

5.0 Summary

In this unit, you have learnt that:

- **KEE** Knowledge Engineering Environment. Frame-based expert system
- A frame is a data structure with typical knowledge about a particular object or concept.
- **Terms Associated with Frame Base Expert System include:**
    - **Frame:**A data structure with typical knowledge about a particular object.
    - **Slot:** A component of a frame in a frame-based system that describes a particular attribute of the frame.
    - **Attribute:**A datum associated with an entity.
- In the context of frames, inheritance refers to:
    - The inheritance by a class-frame of the characteristics (slots) of all its superclasses.
    - The inheritance by an instance-frame of the characteristics of its class-frame.

6.0 Tutor Marked Assignment

- Define the Frame based expert System
- Explain the various terms associated with frame based Expert system.
- Discus KEE and its components

7.0 Further Reading And Other Resources

Bobrow, D.G., and Collins, A.M., Eds. Representation and Understanding. Academic Press, New York, 1975. Includes papers describing early work on frame-based representation languages.

Findler, N.V., Ed. Associative Nekoorks: Representation and Use of Knowledge by Computers. Academic Press, New York, 1979. Contains papers focused specifically on semantic networks.

IEEE Computer Society. Computer, Special Issue on Knowledge Repre-sentation, 16, 10 (Oct. 1983). Provides a representative sampling of recent work in knowledge representation.

Brachman, R. J. (1979). On the epistemological status of semantic networks. In Findler, N. V., editor,

Associative

Networks: The representation and Und erstan din g in Computers , pp 3-50. New York: Academic Press Inc.

Brachman, R. J., Fikes, R. E. and Levesque, H. L. (1983). KRYPTON: A functional approach to knowledge representation. *IEEE Computer* , 16, pp. 67.
Brachman, R. J. (1985). 'I lied about trees'. *AI Magazine* , vol. 6, pp 80-93.
Chen, C., Kuljis, J., Paul, R. J. (2001). Visualising latent domain kno wledge.

IEEE Trans, Systems, Man and Cybernetics – Part C: Applications and Reviews , 31(4), pp. 518-529.
Crowther, P., Hartnett, J. (1997). "Eliciting knowledge with visualisation – instant gratification for the expert image classifier who wants to show rather than tell", Proc. of Annual Conf. of Geo-Computation '97 and SIRC '97. Univ. of Otago, NZ, pp 15-23.
Devaraj, D., Murthy, T. V. S. L. N., Yegnanarayana, B. (1999). "A fuzzy system model for plant condition monitoring", *Proc. Intern. Conf. Computer Modeling, Simulation and Communication (CMSC '99)* .
New Delhi: Tata McGraw-Hil l, pp. 210-214.

Unit 4 Expert System Shells

1.0 Introduction

In the last unit, you learnt about Frame based expert system as well as various terms associated with it. In this unit, you will be exposed to Shells and its components

2.0 Objective

In this unit, you should be able to:

- Define Shells
- Explain the components of shell

3.0 Overview of Shells

A shell (or inference engine) is a complete development environment for building and maintaining knowledge-based applications. It provides a step-by-step methodology, and ideally a user-friendly interface such as a graphical interface, for a knowledge engineer that allows the domain experts themselves to be directly involved in structuring and encoding the knowledge. Examples of shells include Drools, CLIPS, JESS, d3web, G2, eGanges, and OpenKBM (initially developed as a replacement for G2).

Expert system shells provide methods of building expert systems without extensive knowledge of programming through mechanisms that (1) input the decisions, questions and rules that are followed (2) construct a knowledge database that can be manipulated by subsequent parts of the system (3) verifies possible violations of surface validity and (4) operates the "inference engine" that operates on the rules, poses the questions to the users, and determines whether a particular decision is valid (Trollip and Lippert, 1987). Most expert systems also allow the user to halt the processing at any time to query the system why a question was asked, or how a decision was reached (Trollip and Lippert, 1987).

Most expert system shells can now run easily on most current micro-computers and are able to handle the manipulation of a relatively large number of rules and associated questions.

Expert system shells a re expert system development tools consisting essentially of the expert system without the knowledge base, embodying the inference engine, working memory, and the user interface (Sener, 1991). An example of the inference engine part of an expert system that deduces new conclusions from known facts is illustrated below (Sener, 1991):

IF liquid limit=known
AND plastic limit=known
AND plastic limit>liquid limit
THEN soil=non plastic

Expert systems give advice or solve problems by drawing upon this knowledge stored in the IF/THEN rules.

An Expert system shell is a software development environment. It contains the basic components of expert systems. A shell is associated with a prescribed method for building applications by configuring and instantiating these components.

3.1 Shell components and description

The generic components of a shell : the knowledge acquisition, the knowledge Base, the reasoning, the explanation and the user interface are shown below. The knowledge base and reasoning engine are the core components.

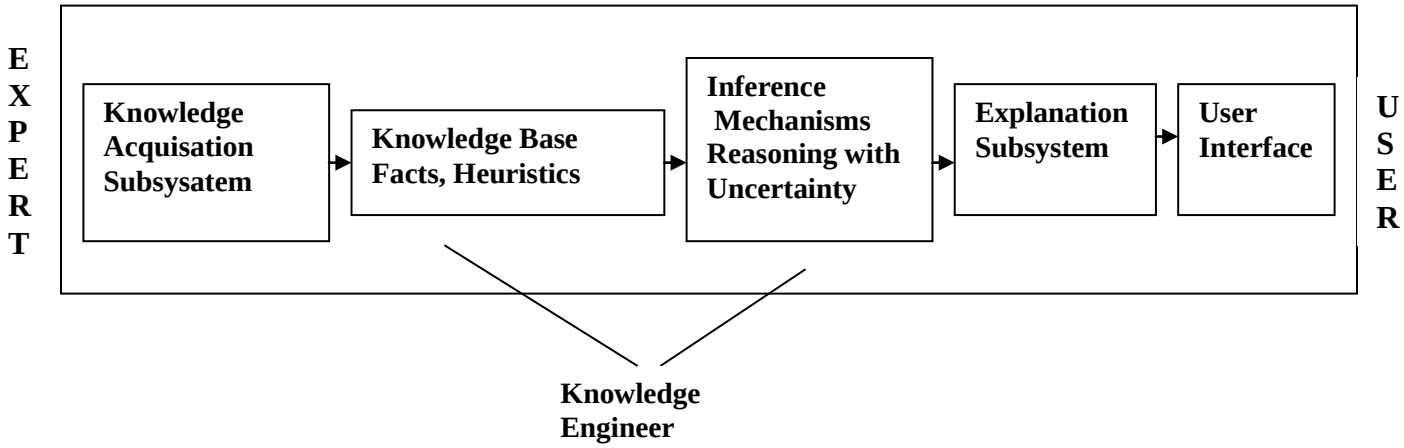


Fig 16: Shell Component
Source:

3.1.1 Knowledge Base

It is a store of factual and heuristic knowledge. Expert system tool provides one or more knowledge representation schemes for expressing knowledge about the application domain. Some tools use both Frames (objects) and IF-THEN rules. In PROLOG the knowledge is represented as logical statements.

3.1.2 Reasoning Engine

Inference mechanisms for manipulating the symbolic information and knowledge in the knowledge base form a line of reasoning in solving a problem. The inference mechanism can

3.1.3 Knowledge Acquisition subsystem

A subsystem to help experts in build knowledge bases. However, collecting knowledge, needed to solve problems and build the knowledge base, is the biggest bottleneck in building expert systems.

3.1.4 Explanation subsystem

A subsystem that explains the system's actions. The explanation can range from how the final or intermediate solutions were arrived at justifying the need for additional data.

3.1.5 User Interface

A means of communication with the user. The user interface is generally not a part of the expert system technology. It was not given much attention in the past. However, the user interface can make a critical difference in the perceived utility of an Expert system.

3.1.6 Explanation

Most expert systems have explanation facilities that allow the user to ask questions - why and how it reached some conclusion. The questions are answered by referring to the system goals, the rules being used, and existing problem solving. The rules typically reflect empirical or "compiled" knowledge. They are codes of an expert's
rules of thumb, not the expert's deeper understanding.

Activity H: What is a shell?

4.0 Conclusion

In this unit, you have learnt about shell and its various components

5.0 Summary

In this unit, you have learnt that:

- A shell (or inference engine) is a complete development environment for building and maintaining knowledge-based applications.
- The generic components of a shell are the knowledge acquisition, the knowledge Base, the reasoning; the explanation and the user interface are shown below. The knowledge base and reasoning engine are the core components.

6.0 Tutor Marked Assignment

- What is a shell?
- Discus the various components of shells system.

7.0 Further Reading and Other Resources

Artificial Intelligence", by Elaine Rich and Kevin Knight, (2006), McGraw Hill companies Inc., Chapter 20, page 547-557.

2. "Introduction To Artificial Intelligence & Expert Systems", by Dan W Patterson, (2007), Prentice-hall, page 1-464.

3."Expert Systems: Introduction To First And Second Generation And Hybrid Knowledge Based Systems", by Chris Nikolopoulos, (1997), Mercell Dekker INC, page 1-325 .

4. "Artificial intelligence and expert systems for engineers", by C. S. Krishnamoorthy, S. Rajeev, (1996), CRC Press INC, page 1-293.

5. Related documents from open source, mainly internet. An exhaustive list is being prepared for inclusion at a later date.

Module 3 A Group Proposal Submitted on Expert System.

Unit 1 The Expert System as a Proposal for Creating Innovative Strategy

Lendel Viliam, Varmus Michal

1.0 Introduction

The current period is marked by many changes. Competition is constantly sharpening, businesses feel effects of global economic crisis, constantly developing new products, customers, primarily through the Internet have a lot of information on which the decision to buy. Cus- tomer focus has become part of a corporate approach of many companies (Strišš et al., 2009). Efforts to identify customer requirements and transferred to the new product is the task of today's marketers.Businesses are trying to exploit these changes and turn them into their own competitive advan- tage. Innovation activities are essential to the survival and growth of the company. They signifcantly affect to its performance, market position and market power. They are a key business process, because through these companies are trying to achieve some competitive advantage.The basic precondition for the creation and use of innovation in the enterprise is well worded

2.0 Objectives

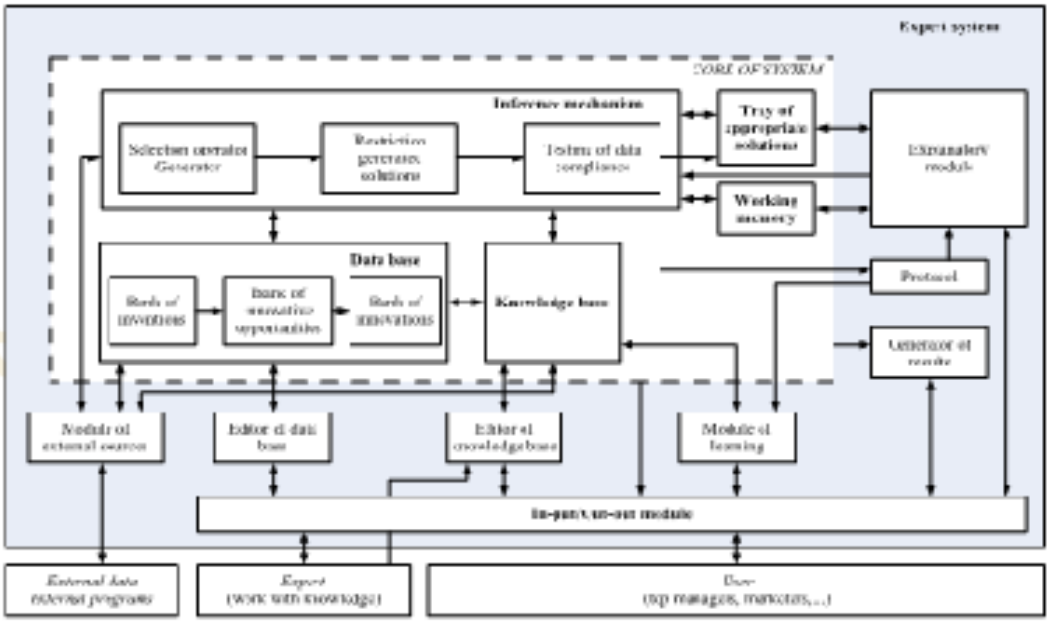
The main objective of the proposed expert system will achieve the best response to the real data on innovation, thereby ensuring high quality decision-making innovation strategy. Creating an expert system is a complex process as the project site, as well as for programming.

3.0 The Components of the Proposed System

Thec Authors Proposed that for an expert system to work with the knowledge needed to create an innovation strategy, the following basic parts should be in place:

- The core system (knowledge base, data base, working memory, and stack mechanism infer- ence appropriate solutions),
- Input / output module,
- Explanatory module,
- Protocol
- Other components of the system (knowledge base editor, editor of the database module learning outcomes generator module external sources. identifcation of key elements, work with innovative ideas, opportunities, innovation and ap- plication of lateral thinking. Research was conducted on the sample 236 senior managers of medium and large enterprises operating in the Slovak Republic. Most managers were contacted

through an electronic questionnaire (93.2%). 6.3% of top managers have personally reached out through semi-structured interview.



CHARACTERISTICS OF FUNDAMENTAL ELEMENTS OF THE PROPOSED EXPERT SYSTEM

The proposed expert system consists of modules, which provide its functionality. Each module performs a specifc task. Outputs from one module are inputs to the second module. The successful performance of the proposed expert system is essential to ensure coherence and seamless communication between modules.

Knowledge Base

Knowledge Base focused on expert knowledge gained. It provides a space for the collection of all knowledge that can be used in the innovation process. Tidd (2007) points out that innovation is closely linked with knowledge. It states that "the issue is to create new opportunities by combining different sets of knowledge" (Tidd, 2007, p. 16). These are the knowledge of technological possibilities and the appropriate configurations to meet the needs of business and customers. This knowledge may take the form of experience accumulated by the company during its existence or be the result of the review process (technology, market competition ...). The main purpose of knowledge base is to provide space for an appropriate mix of skills into a successful innovation.

In addressing the knowledge seeking knowledge manager needs to proceed further in solving the problem. The knowledge base must be designed to allow efficient access to required knowledge, while allowing the largest store of knowledge. It must correspond to the choice of implementing data structures.

Data Base

Data Base contains all the unique information relating to innovation. It consists of Bank of inventions, the Bank of innovative opportunities and the Bank of innovations.

Bank of inventions is the search space, creation, evaluation and storage of inventions that may participate in the next phase in creating an innovation strategy. Invention is new knowledge that results in a change in the structure and level of knowledge and this is a new idea of possible change, respectively new approach (Zaušková, 2006, p. 37). It is about suggestions, ideas and thoughts of a solution. The process of inventions is a complex process that requires a large number of variants. Innovative ideas are the product of information coming from the surroundings and the business system itself. This information is identifed to the needs and concept solutions. Important processes are the recognition of status and choice of appropriate means. The result of this process is an innovative idea (invention), which is ready for their review and decision on its need and value for the purposes of innovation strategy. Bank of innovative opportunities is a space to store and work with the identifed innovation op- portunities. An innovative idea (invention) is a necessary but insuffcient condition. The idea must be feasible and satisfy the conditions for potential success. We are talking about fnd of opportunities - the process of change an idea into an opportunity. Review inventions opportunity for innovation is a complex process that requires some assessment of up to 100 ideas for one successful product (Zaušková, Loučanová, 2008, p. 40). Here, the enterprise can create its own rating system. During evaluating, the managers must reckon with feedback in the form of corrections and improvement ideas, combining them, and breaking down under (Zaušková, Loučanová, 2008, p. 40). The Bank also serves innovative opportunities for the conservation of innovative opportunities for the company are not immediate importance. Bank to gradually accumulating innovation opportunities for peer evaluation can create the necessary synergies.

Bank keeps all the innovation created by innovation and creates the environment for their effective management and their translation into successful innovation strategy. Innovation is understood the practical transfer of new ideas into people's products, services, processes, systems and social relations.

Working memory

Working memory provides space for the solution of the problem. It will include three components: a solution, plan and agenda. Solution provides to the space in working memory, which requires the solution to the problem created. It is about hypothesis, proposed structure and so on. The plan contains a sequence of actions to be taken to solve the problem. The plan may, for example a sequence of such steps: design the scheme-specifc variants of innovation strategy and then optimize. The priorities of each decision module plans - scheduler, which selects the plan and the work to be carried out within the plan developed. Agenda in the form of data structure stores a sequence of actions to be taken to implement the selected activities planner. The priorities assigned to interpret these actions to maintain and ensure the activation of broken shares in the event that other actions have higher priority. Working memory can be seen as a place where solutions are stored intermediate data and arrange the activities of the solutions.

Input-Output Module
The main task of I / O module is to create an interface between knowledge-based system and its surroundings, which is represented by end-users. These are the senior managers and staff involved in the process of creating an innovation strategy. Other end users are business marketers who use the knowledge gained in the evaluation of inventions and innovative opportunities in the implementation analysis to create an innovation strategy. Through this module, the user sees an expert system. The main objective in this area is to create user interfaces that allow communication affordable for solving the problem, i.e. while creating an innovative business strategy and interpretation of appropriate solutions found.

Explanatory module
The basic module is a function of clarifying clarifcation, explanation and justifcation of the decision, which is the output of an expert system. Managers and marketers obtain the necessary justifcation for the fnal solution in the form of the chosen innovation strategy. Explana- tory module offers managers and marketers the opportunity to browse the knowledge base, view and modify working memory. They get invaluable help when debugging a knowledge base to guide the course of the solution. The importance of clarifying the module and see the possibilities of development of various innovative options strategies. Managers would be able to choose alternatives to develop innovative strategies that expert system still has not been examined. Explanatory module would allow examining the impact of certain decisions on innovation located by the innovation strategy. History of the solution is charged in a protocol module then uses to explain.

Inference mechanism
It is allows fnding the required knowledge in knowledge base, data base and using them to develop innovative strategies. It can derive from these bases for further information and knowledge. Its work is based on a knowledge base and data base on which infuences the choice available to operators, limiting the number of tested solutions proposed generator and controls compliance testing solutions generated with the actual data. One of the important outcomes of inference mechanism is tray of appropriate solutions. This module contains the appropriate solutions, which are rated according to their ftness level. They then enter into explanatory module, through input / output module which gives to user (the senior manager or marketers).

Other components of the proposed expert system
Other important components of the proposed expert system are:

- Editor of knowledge base
- Editor of data base,
- Module of learning,
- Generator of results
- Module of external sources.

Editor of knowledge base provides a constant update, completion and dissemination of knowledge base. This change occurs not only during its development, but often also during operation.The reasons leading to these changes may be different. The most common is the acquisition of new knowledge that may help in the process of innovation strategy of company.

Second, the manager can identify errors that must be removed (such as rules for generating variants of an innovation strategy, importance of amending the innovation process ...). The same principle is based on the editor of data base, which, unlike the editor working from knowledge base information relating to inventions, innovative opportunities and innovations themselves. There may be reason for changes such as incorrect identifcation of ideas and their subsequent translation into innovative opportunities...An important part of the proposed expert system is a module of learning. Its main objective is to promote acquisition of knowledge. It ensures that is always based on the current situation.The obtained knowledge is stored back into a knowledge base and used for future proposal innovative business strategy.Generator summarizes the results of partial results in a reasonably integrated whole, without extra information requested in the form of a comprehensible form. His contribution is in providing effective, effcient, ifferentiated, comprehensive, current, serving mainly commercial information for decision-making in innovation. Communicates with input / output module to provide senior managers with the required information and understandable.Module of external sources provides communication of the expert system with their environ- ment. The main activity of this model is to work with external data and work with external programs. Inference mechanism in case of request certain data will search in the data base. If there in not fnd the required information, the management is submitted to module external sources. He begins to search external data source. In case of found of required fact it is allowed insert it into the data base and send back the management to inference mechanism. Likewise it does even in case of necessary expertise. As inference mechanism doesn't appear requisite knowledge in the knowledge base, submit the management to module of external sources. It begins to search external data source. If successful, will embed the acquired knowledge into a knowledge base, if fail then it turns through the input / output module for expert, who knowledge supplemented by the necessary knowledge base editor. Then handed back control to inference mechanism.

CONDITIONS FOR SUCCESSFUL WORK OF THE PROPOSED EXPERT SYSTEM

That the proposed expert system to work effectively, it is necessary to ensure that the follow- ing conditions:

- Effcient data acquisition
- Implementation of quality input / output module
- A detailed and careful analysis of the enterprise environment.

The quality of the implementation of input / output module can substantially multiply the performance generated by the system (Návrat et al., 2007, p. 259). These are the creation of executive comfort interactions with expert systems. The aim should be to create an acceptable user interface allowing the creation of innovative communication strategies and appropriate business located as a result the interpretation of the innovation strategy.

The most important prerequisite for the successful operation of an expert system we consider a detailed and thorough analysis of the enterprise environment. In a frst step, the undertaking must determine its innovation capacity. It consists of the sum of knowledge, experience, resources, assets and managerial capabilities and skills in business available, or is unable to obtain in time.

This is the basis for creating an innovation strategy. Then there is a mapping of innovation potential, the rate of innovation means business, it can reach the optimal utilization of all components
of innovative capacity. The next step, the enterprise must assess and identify the current level of use of innovative capacity. This analysis will provide a realistic picture of the possibilities of innovation, which in turn translate into specifcation of innovative requirements. These are the election of the main operators, i.e. areas in which are interesting for the enterprise in terms of its visionand mission and will form the essence of innovation strategy. It can be the innovative area in which a company makes in terms of innovative capacity using the best results. Another area that needs to be addressed is the establishment of rules making innovation strategy. The rules will operate the proposed expert system. An important part of innovation is to defne requirements, system of evaluation. The company must have clear criteria by which consider innovation strategy chosen, respectively according to which attributes to monitor its implementation over time. These attributes will form the basis for continuous evaluation of innovative strategies that will indicate the timeliness of an innovation strategy and the measures for its upgrade.