Chapter 6 - Expert Systems and knowledge acquisition

An expert system's major objective is to provide expert advice and knowledge in specialised situations (Turban 1995). ES is a sub-discipline of AI (Turban et al 2001). For an ES to reason, provide explanations and give advice, it needs to process and store knowledge. Knowledge in the form of facts and rules are kept in a knowledge base.

6.1 Expert System definitions

ES definitions by Turban et al (2001) and Olson & Courtney (1992) were presented in Paragraph 2.3.1 (p19). Raggad & Gargano (1999) state: "ES emerged, as a branch of AI, from the effort of AI researchers to develop computer programs that could reason as humans". Goodall (1985) distinguishes two approaches when defining an expert system:

- The human/AI oriented approach, and
- The technology oriented approach

Goodall's (1985) *AI approach* defines an expert system as a computer system that performs functions similar to those normally performed by a human expert. He extends this definition to include the nature of an expert system, in other words, he includes how ES behaves, in the *technology approach*:

> "An expert system is a computer system that uses a representation of human expertise in a specialist domain in order to perform functions similar to those normally performed by a human expert in that domain".

In formulating the above definition, Goodall (1985) focused on the implementation of two types of ES:

- ES performing at a suitable 'high' expert level, and
- ES that performs tasks which no human expert has, using perhaps a very complex set of rules to control a chemical process at a speed and in circumstances (e.g. temperature, time, location) which no human could possibly match. Such a system will most surely outperform the best human expert.

The value of ES as a sub-discipline of AI (Turban et al 2001), can be appreciated more when comparing AI to natural intelligence and conventional programs. Kaplan (1984) as referred to by Turban et al (2001), states several commercial advantages of **AI over natural intelligence**:

- AI is more permanent. Employees take knowledge with them when they leave their place of employment, or worse: an employee may even forget his knowledge. As long as the computer systems and programs remain unchanged, the knowledge remains the same.

- AI offers ease of duplication and dissemination. Transferring knowledge between persons may be a lengthy process. Some expertise is never transferred. Knowledge embodied in a computer system can be easily moved to another computer.
- AI can be less expensive than natural intelligence. Buying computer services can cost less than having a human performing the same task.
- AI is a computer technology and thus consistent and thorough. Natural intelligence is erratic because people are erratic and perform inconsistently. , and
- AI can be documented. Decisions made by a computer can easily be documented by tracing the activities of the system. Natural intelligence is difficult to document. A conclusion reached may be difficult to be recreated or recalled later. Some of the assumptions or even the reasoning process may be forgotten.

Natural intelligence has several advantages over AI:

- Natural intelligence is creative: In an AI system, knowledge has to be carefully constructed, while knowledge is inherent to all human beings
- Natural intelligence uses sensory experienced directly: Users using AI, use sensory experiences indirectly
- Natural intelligence enables people to recognise relationships between artefacts, sense qualities and spot patterns that explain how various items interrelate, and
- A wide context of experiences is used when natural intelligence solves an individual problem, while AI typically gains its power by having a very narrow focus

ES differs from conventional programs too. Raggad & Gargano (1999) and Turban et al (2001) summarise the **characteristics of conventional programs** as follows:

- Conventional Computer Based Information Systems (CBIS) *processes data and information* as a direct conceptual resource
- Conventional programs are based on *algorithms* or mathematical formulas and sequential procedures that lead to a solution using data to solve problems
- The *objective* of CBIS is to create and efficient solution by designing an efficient algorithm
- CBIS are characterised by *repetitive* processing
- The processing system or *control* mechanism of the system and the knowledge of the subject are *intertwined* as part of the sequence of instructions
- *Knowledge of the subject is built into the processing mechanism* or control of the system. The subject knowledge or business rules form part of the sequence of instructions. All business rules are part of the program code. The knowledge is stored as control embedded in the program instructions. The control is the sequence in which the transactions are performed. The programmer fixes these steps in advance. It is called the algorithm of the program.
- To change the business rules or domain *knowledge* a programmer needs to edit the program and insert or change the statements that represent the information *in the correct place*

- The program or system used *does not explain which rules* were used during processing of data to produce the output
- The program is specialised to deal with specific information. A *complete system* of programs and input are *necessary* to achieve something useful.
- *One missing essential step causes the program(s) to malfunction*. The data that are processed should be complete and certain to achieve an appropriate result or valid output.
- *Error detection* of the domain knowledge is performed *when executing the program* and noticing faulty output
- CBIS frequently uses *algorithms* in its search approach ,and
- CBIS has no *reasoning* capability except for a fixed process in the algorithm of the program, and
- *Updating and maintenance of data and information are usually difficult*

When with conventional programs, **the characteristics of ES** are:

- The main conceptual source of ES is *knowledge*. ES can expand to include a knowledge acquisition component that processes data and information into rules. Data and information are indirect conceptual resources.
- ES does not represent its knowledge in algorithmic-like procedures. AI is based on *symbolic* processing of knowledge. Knowledge is represented by symbols such as letters, words or numbers that represent objects, processes and their relationships. The knowledge base contains facts and concepts and relationships among them. Processes are used to manipulate the symbols in the knowledge base to generate advice or recommendations to solve problems.
- The *objective* of ES is to create an efficient knowledge representation to support the inference mechanism
- ES keeps the *control separate* from the rules: the inference engine is responsible for most of the control. The knowledge base contains the rules or knowledge.
- *Knowledge is expressed as data* and not encoded in the control of the ES. This has the advantage that knowledge can be returned to the user as explanations of the ES conclusions.
- The *order* of the rules in the knowledge base *does not matter*
- ES *provides an explanation facility*
- The rules that comprise the knowledge may be built bit-by-bit, a rule at a time. ES can *tolerate imperfect, incomplete or uncertain business rules* or domain knowledge.
- The ES will *produce results* even if the knowledge base is incomplete and uncertain. This is because of the separation of the knowledge and control components.
- ES frequently uses *heuristics* in its search approach
- ES has a *reasoning* capability, and
- *Maintenance and updating of data are relatively easy*. Changes can be made in self-contained modules.

6.1.1 When to use an Expert System

ES is appropriate for a limited number of problem domains. The task should be well bounded and should not include a large number of event combinations. A general rule of thumb is that if the average employee can solve the problem in a few minutes it is not worth solving it by an expert system.

ES is appropriate (Raggad & Gargano 1999; Mallach 1994):

- If great costs are involved when making the decision
- When applied to repetitive problem domains, the task must be performed often
- When a big difference exists between the best solution and the worst solution
- When test data is easy available to test and validate the ES
- When there is a general agreement on the system's conclusions. Errors in the input should be tolerated in the system output. Experts must agree on the solutions.
- When recognised expertise is available and a necessity during the development of the ES
- If the problem is clearly specifiable in the area of expertise i.e. what the system is supposed to do before it is done, the desired system can be developed
- If the problem is identifiable with a human expert, that is based on human expertise. Experts must perform the task substantially better than non-experts. The human expertise must be scarce.
- If the problem domain is well bounded: for example defining criteria to determine the subject matter within the system from the matter outside of the system: The task is reasonably stable and within an acceptable narrow domain.
- If solving the problem is based on knowledge rather than common sense reasoning: ES is based on knowledge and naïve systems on reasoning.
- The solution has to be possible, justifiable and concisely generated (using only a few hundred rules), and
- The knowledge has to be of a cognitive style and independent of common sense. Physical activities are ruled out.

Goodall (1985) provides a comprehensive list of suitable types of tasks appropriate using ES in solving problems that includes:

- Interpretation
- Prediction
- Diagnosis
- Debugging
- Design
- Planning
- Monitoring
- Instruction, and
- Control

Turban et al (2001) calls the above-mentioned types the generic categories of Expert Systems. Most ES systems will include one of the above problem areas as a dominant characteristic, but may include some aspects of the others.

6.1.2 Advantages and disadvantages of using ES

Developing an ES is time consuming for the expert, whose knowledge is captured, the knowledge workers, who will eventually make use of the system; as well as the programmers, who will be involved in the building of the ES. ES is limited in the sense that it focuses on a narrow problem domain and can not therefore be applied outside of the specific problem domain. For this reason it involves intensive development cost.

According to Goodall (1985), Mallach (1994), Turban (1995) and Turban et al (2001) the advantages of using an expert system (a system with intelligent reasoning) above a conventional information system (for example a system without intelligent reasoning) are:

- It improves the quality of the system. The program will function even with incomplete and uncertain data from the very first time it executes previously non-programmable tasks. This is an advantage in domains where knowledge changes frequently. It provides consistent advice whilst reducing the error rate. More rules can be added to the existing knowledge as knowledge about the domain increases.
- It can handle uncertainties expressed as probabilities. The way it is handled depends on the inference engine of the particular ES.
- It explains the logic behind its recommendations, making the knowledge explicit
- It can be used as a training vehicle for users who lack the expertise
- It provides monetary savings once implemented. ES does not cost money when not being used. Costs can be cut as the human expert's availability is not needed all the time.
- Experts are freed enabling them to focus on tasks requiring their expertise
- It can codify and preserve knowledge of the specific problem domain. It offers a solution to scarce expertise where few experts exist for a task or where the expert is about to retire or leave the job.
- It increases programmer productivity. The author of the knowledge base will have access to more advanced techniques and tools than the system's eventual user. Separating the expert's knowledge into modular rules reduces the change of development errors and improves the maintainability of the system.
- It can discover new knowledge
- It can provide increased output and productivity. ES works faster than humans do. Increased outputs mean fewer workers and reduced costs. ES can be replicated as needed.
- It eliminates the need for expensive equipment used by human experts for monitoring and control
- It makes knowledge and information accessible, and

- It can outperform human experts because of the fact that ES:
  - Make fewer errors
  - Do not become tired or bored and never sleeps.
  - Will not overlook a solution
  - Can handle large volumes of data
  - Can respond more rapidly
  - Can function in hostile environments such as deep-sea drillings and reactor control, and
  - When the knowledge of several experts is integrated, it can outperform any one expert

Another advantage of ES is that it is a tool for manipulating knowledge. It can:

- Discover new knowledge e.g. discovery of a new prime number
- Codify and explicitly state old knowledge by forcing the author of the knowledge base to formalise the expertise in rules or other structures. This resultant knowledge can be used by anyone who needs it. Tools are available to codify knowledge. One such tool is Teiresias.
- Assist in teaching once knowledge is codified, and
- Help analyse knowledge

Mallach (1994), Turban (1995) and Turban et al (2001) give the drawbacks of ES as:

- The domain of expertise is usually narrow. An ES is designed for a specific purpose and is not useful for another purpose.
- ES cannot apply common sense, only its rules
- Experts do not realise when they reach their limits. They may recommend inappropriate actions. The experts' performances "falls off a cliff" rather than degrade gradually.
- ES may be costly to develop because of the time of human experts and other people involved in the process
- Knowledge is not always readily available
- Expertise is hard to extract from humans
- Lack of trust by end-users may be a barrier to ES use
- Knowledge transfer is subject to a host of perceptual and judgemental biases
- The work of rare and expensive knowledge engineers is required, and
- Most experts have independent means of checking whether their conclusions are reasonable

6.2 Architecture of ES

6.2.1 The process of ES

Transferring expertise from an expert to the computer and then to the user involves four activities: knowledge acquisition, knowledge representation, knowledge inference and knowledge transfer (Turban et al 2001). Knowledge is acquired from experts and other documented sources and then represented or organised as rules or frames. These rules or frames are electronically stored as a body of knowledge or a knowledge base. This knowledge base forms one of the distinct parts of an ES. The

reasoning mechanism that draws conclusions from the knowledge attained in the knowledge base and the knowledge obtained from the user forms the other distinct part of the ES. This reasoning mechanism is known as the inference engine. The inference engine results in advice or recommendations for novices. When understood by the novice, possibly by accessing the explanation given by the ES, knowledge has been transferred to the user. Figure 2-6 (p21) shows the different components of an ES and their interaction.

6.2.2 The components of ES

An expert system consists of various major components (Goodall 1985):

- A dialogue structure
- A knowledge base, and
- An inference engine

Turban et al (2001) adds:

- A blackboard
- An explanation sub-system, and
- A knowledge refining system

The **user interface** provides the conversation of the system with the user. It is responsible for posing the questions to the user, reading the user's reply and explaining the rules used to reach a conclusion. A survey of typical ES suggested that on average (Goodall 1985):

- Eight percent of the code is used for the inference engine
- 22% of the code is used for setting the knowledge base, and
- 44% of the code is used conversing with the user

The **explanation capability** is one of the most important features of an ES and it realises in the user interface. Turban et al (2001) views the explanation facility as a separate component of the ES that can trace and explain the ES' behaviour (See Figure 2-6: p21). The body of knowledge (also called the **knowledge base**) and the **inference engine**, the reasoning mechanism of the ES, form the two fundamental parts of the ES. These two subsystems provide the ES' main functionality. The **knowledge base** represents domain specific knowledge in a specific form. The **inference engine** deduces facts or draws conclusions from the knowledge base based on the user input and the facts from the knowledge base and/or other sources. The inference engine and the knowledge base exist as two separate modules of the ES that work closely together.

♦ The user interface

The knowledge worker (also called the user or decision-maker) gains access to the ES via the **dialogue structure** provided by user interface. The knowledge worker provides the ES with input and receives the system's explanation of how it reached its conclusion. Output to the user usually comprises of taking text from the knowledge base and slotting them into a few predefined sentence formats.

When allowing the user to use **natural language** in obtaining input, a misunderstanding could result because of the ambiguous characteristic of natural language. The use of natural language in the dialogue with the user relies on the meaning of the user's utterances. The meaning of what exactly is meant lies in examining the surrounding context and is prone to be misunderstood. The naïve user is likely, by the system's ability to read natural language, to be fooled into thinking that the ES absolutely understands what he (the user) means. The best way to solve this problem is for the ES to generate all possible interpretations of any sentence that could be ambiguous, feed it back to the user, and enquire of the user which interpretation he (the user) really meant. The easiest way to understand the user is not to allow conversation to be in any natural language, but rather via a **predefined syntax**, requesting **preformatted specific input**. Questions posed to user and explanations of conclusions that are reached can be given by taking the text provided by the author of the knowledge base and slotting it into a few predefined sentence formats.

♦ The knowledge base

The power and effectiveness of the ES is equal to the quality of the knowledge it contains. The knowledge has to cope with high degrees of complexity and apply the best judgement. The acquiring of expert knowledge is crucial and involves the gathering of information about a domain usually from an expert. This information is incorporated in a computer program stored as a knowledge base. Obtaining knowledge from humans can be a difficult task. Hayes-Roth et al (1983), as referenced by Turban (1993) and Klein & Methlie (1995), views **knowledge acquisition** (See Paragraph 6.4: p122) or **knowledge engineering** as being composed of five stages:

- Identification or definition of the problem and the major characteristics of the problem
- Conceptualisation or the choice of an appropriate representation medium such as rules and the acquiring of the knowledge associated with the certainty of the knowledge
- Designing of an architecture or deciding the formalism or acquisition methodology e.g. rules and the deciding the process extracting knowledge from the experts
- Implementation, building or the programming of knowledge and the development of a prototype, and
- Testing and refining of the knowledge base by subjecting it to examples, examining the validity of the knowledge.

The *methods* of knowledge acquisition can be divided into *manual*, *semi-automated* and *automated*. The primary manual approach is interviewing, ranging from completely unstructured to highly structured interviews. Automated knowledge acquisition uses an induction system with case histories and examples as input to derive the knowledge base. It eliminates the role of the knowledge engineer and expert (Turban 1993). Automatic knowledge acquisition is also known as machine learning. Knowledge collected must be analysed and coded prior to its representation in the computer. Automated knowledge acquisition methods are easier to validate and verify. On average fewer knowledge mistakes occur when acquiring knowledge from multiple experts because of the enhanced

synergy among experts. Difficulties such as meeting time scheduling and resulting compromising solutions in opinion conflicts may arise.

Knowledge representation is important and crucially affects the ease and speed with which the inference engine can use it. Knowledge representation implies a systematic means of encoding what an expert knows about a knowledge domain in an appropriate medium (Goodall 1985). A number of knowledge acquisition techniques have been developed. Turban (1993) discussed a variety of techniques. The selection of a technique depends on the types of knowledge that should be retained in the knowledge base. Knowledge can be classified as **surface knowledge**, to put declarative and procedural knowledge into heuristics to solve a problem quickly; or **deep knowledge**, which involves fundamental knowledge of domain including the definition, axioms, general laws, principles and causal relationships upheld by the knowledge. Surface knowledge is the basis for most common ES using production rules. Production rules are widely used and quite efficient in diagnosis problems. They are used to encode rules of thumb also called heuristics or knowledge used by humans (Turban 1995). Knowledge can be formulated using formal theories or normative models.

Knowledge representation is the systematic means of encoding knowledge of the human expert in an appropriate medium (Beynon-Davies 1991). Knowledge can be represented as:

- Predicate calculus or formal logic
- Business applications in the form of production rules
- Semantic networks, which organise knowledge through nodes in a graph rather than data structure and represent relationships between the facts by links between the nodes, and
- Frames or structured objects that use data structures to store all structural knowledge of a specific object in one place

Logic itself is not a way for computers to store knowledge, but proves to be a vital tool to think about how computers store knowledge. Logic is part of mathematics and can be used in various forms to reason about the correctness of computational representation and inference (Goodall 1985). The forms of logic include:

- Programming languages such as PROLOG (programming in Logic)
- Pro-positional logic or calculus that consist of building blocks such as elementary sentences, joined by ‘and’, ‘or’ and ‘not’. The internal structure of the elementary sentence is irrelevant. , and
- Predicate logic with its basic building block objects and relations such as “is-a” and “has-a” between them to build statements. The relations is called predicates and deals with the logical operators “and”, “or” and “not”.

The above provide a theory to formalise the study of reasoning, determining valid knowledge representations. It is used to prove the correctness or determine that certain types of inference are correct or incorrect.

The forms of knowledge representation often used in ES are:

- Rules
- Semantic nets
- Frames, and
- Cases

Rules are often called production rules and the program that reasons with rules a production system, especially when the inference is data-directed forward chaining. Most ES represent knowledge as rules and therefore the knowledge base is often referred to as the rule base. The reason for its popularity is that almost every piece of knowledge can be written as a rule. Many ES exist that requires rules as input and tempts knowledge workers to express knowledge as such. Rules are natural and the only way to codify some knowledge. Rules are a simulation of the cognitive behaviour of human experts. It represents knowledge, but also represents a model of actual human behaviour. Rules are easy for a human expert to read, understand and maintain. If the knowledge is expressed as data and not encoded in the program's control mechanism, it can be returned to the user in the form of explanations. Production rules involve simple syntax that is flexible and easy to understand. They are quite efficient in diagnosing problems of the form:

- if (condition)
- then (conclusion)

Each production rule in a knowledge base implements a chunk of expertise and when fed to the inference engine, as part of a set, should synergistically yield a better result than any of the rules individually. In reality, rules are highly independent and adding a new rule may contradict other rules or cause other rules to be revised. Rule systems can broadly be classified into simple, all rules on the same level and available to every search cycle, and structured rule-base systems where searches are limited to segments of the rule base, thus improving the efficiency or the search. A rule set is a named collection of individual rules pertaining to a distinct aspect of a problem and helps in comprehending and maintaining the rule base. This structure is a kind of meta-knowledge that is imposed on the knowledge base. Each sub problem could have its own rule set (Klein & Methlie 1995).

Semantic Nets is a popular and easy-to-understand way of representing non-rule knowledge. Semantic networks organise knowledge through nodes in a graph rather than a data structure. Links or arcs present relationships between the named nodes. The links or arcs represent relationships such as is-a, has, is, own, needs and reflects the association between concepts. An ES that stores knowledge as a semantic net incorporates an inference engine devoted to operations like inheritance. Such an inference engine will consist of two parts. One part will deal with rules by forward chaining, backward chaining or some other method. The other part will handle net operations matching relevant links in the net to deduce facts.

Objects can be described by a number of features or attributes, many of which stay constant from one instance to the next. A **frame** is a piece of structured data about typical characteristics of an object, act or event. The knowledge is more descriptive than procedural. Similar to rules, frames must be able to deal with uncertainty and missing values. Frames may have default values and slot-filling procedures associated with the slots, to cope with missing values. Frames enable reasoning about object features such as inheritance and the occurrence of exceptions. The reasoning process starts by identifying a frame as applicable to the current situation. Matching the set of frames against the facts available selects an appropriate frame. The use of frames is relevant to non-monotonic logic. Non-monotonic reasoning expresses reasoning with default attributes.

Case-based reasoning is a process that uses similar problems to solve the current problem. It consists of two steps:

- Find those cases in memory that solved problems similar to the current problem, and
- Adapt previous solutions to fit the current problem

The case library forms an extra important component in case-based reasoning. The inference cycle using CBR consists of:

- Retrieving solutions
- Adapting solutions, and
- Testing solutions

The critical step is to find and retrieve a relevant case from the case library. Cases are stored using indexes. The stored case contains a solution, which is then adapted by modifying the parameters of the old problem to suit the new situation resulting in a proposed solution. The solution is tested and if found successful, added to the case library (Klein & Methlie 1995). Knowledge acquisition is easier in case-based reasoning because of the granularity of the knowledge. Knowledge is presented in precedent or resultant cases.

Beyond the knowledge representation language (rules, semantic nets, frames, cases), the knowledge engineer needs further aids such as tools to edit the knowledge base; inference tracers to assist in error detection; and analytical tools to find, update and consistently check the represented knowledge or attributes (Klein & Methlie 1995).

♦ The inference engine

The mechanism that performs the search and reasoning in rule-based systems is called the **inference engine**. The inference engine is activated when the user initiates the consultation session. The inference engine finds the rules that match the given facts, selects which rule to execute and executes the rule by adding the deduced fact to the Working Memory (WM). The inference engine uses pattern matching to select the qualifying rules. The choice of which rule to fire is done by conflict resolution. The most commonly used conflict resolution strategy is the first found strategy. The first applicable

rule is executed (Klein & Methlie 1995) or fired by applying rule deduction or using formal logic. A new fact is concluded and added to the working memory and new patterns found that match the new fact. This sequence of steps and the linking of facts and patterns and rules are known as chaining (Klein & Methlie 1995).

The inference engine deduces facts or draws conclusions from the knowledge base based on the user input and the facts from the knowledge base and/or other sources. Three basic techniques are identified when inferring facts or conclusions from the knowledge base:

- Forward chaining
- Backward chaining, and
- Hybrid chaining: using both forward and backward chaining

New knowledge can be inferred from the existing knowledge in the knowledge base. The specification of how the reasoning is done is the core of the ES. This programming task is independent of the subject knowledge in the knowledge base component. The reasoning method treats facts as arbitrary symbols. The same inferencing method can be used by different kinds of knowledge (Goodall 1985). Various algorithms can accomplish this task. One such algorithm used by ES, is the **Rete Algorithm** (See Paragraph 4.2.9: p45)

Forward chaining match the current state with the rules' antecedents or conditions in the knowledge base. If the condition is true, the inference engine adds the conclusion to the list of known facts. Known facts imply other facts. This technique is also known as data driven. It can be very inefficient, especially if many rule conditions match the data provided by the user. The data provided by the user is examined and new knowledge is concluded or asserted and responded to.

The second technique, **backward chaining**, also known as goal driven or directed, is more efficient if the number of goals is limited and involves the acquiring of information from the user in the form of questions to draw specific conclusions. This style is used to confirm diagnostic tasks and works backwards from what needs to be proved to the facts that might affect it. It is a goal-directed backward chaining of rules to try to prove a hypothesis made by the system. Sometimes stating the conclusions and trying to prove them are unhelpful because none of the multiple goals exist. If this is the case, forward chaining should rather be considered

When a large problem domain is involved, a more efficient program is yielded when the two techniques are used in combination - **hybrid** chaining. To model the strategies used by a human expert, an ES has to employ complex inference that almost always include both forward and backward chaining with potentially many variations. One inference engine will not suit all possible tasks solved by an ES. A typical rule-based system represents heuristic knowledge: that is short cuts in the reasoning procedure. This process uses shallow knowledge.

Model-based reasoning is based on fundamental or deep knowledge of the problem domain. The relationships modelled could be either structural or behavioural. Structural relationship models are used for advice systems and behavioural relationship models for Decision Support Systems that do scenario analysis such as a "what-if" analysis. The behavioural model consists of causal relationships that can be quantitative or qualitative (Klein & Methlie 1995). In model-based reasoning, the reasoning follows the fundamental relationships, the cause-and-effect paths using a deeper knowledge. The reasoning styles are directed by what need to be proved (goal directed) versus by what data are available (data directed). When pursuing the goal directed style and several rules exist that prove some conclusion the rule with the best change of proving it, is selected. This is done by conflict resolution, for example a more specific rule is chosen above a less specific rule.

To simplify the process each rule can be assigned a number to reflect how useful it is. The number could be a guide to select one rule from several that proves the same conclusion. Rules that deal with uncertainty or imperfect data or are referred to as approximate rules, are called **probabilistic reasoning**. An integral part of the working of the ES is how the inference engine determines when to enquire the appropriate information from the user. Probabilistic reasoning is a method used by an ES to draw inferences from the domain and problem knowledge where the knowledge or its implementations or both are less than certain.

Three ways exist to represent uncertain knowledge:

- A single subjective probability is assigned to the proposition, possibly derived using Bayesian inference. Bayes' theorem provides a way of computing the probability of a particular event given some set of observations. The main fact here is not the derivation of the value, but the association of the specific proposition with a single value. Criticism against this approach include that the single value will not reveal much about the rules precision and yet another states that this single value combines the evidence for and against this proposition without reflecting how much of each there is. An alternative approach is to provide an independent measure of belief and an independent measure of unbelief that can be combined into a single certainty factor.
- Fuzzy logic is another technique that can handle inexact data, and
- A final method is the use of a few quantitative qualifiers to express the levels of probability. Possible values may include unlikely, possible, likely, impossible or certain

The knowledge base can include meta-rules. A meta-rule is a rule about the rules and potentially affects the sequence of all other rules called. Meta-rules make the knowledge base more complex and more difficult to understand e.g. the use of different rule sets. Two types of **facts** exist: those received from the user in the form of input data (known facts) and those deduced from the input data (inferred from primitive data). Knowledge can also be obtained from other sources such as databases (DB). Much research is connected to Expert Systems and databases to refrain from transcribing some of the information into rules.

In acquiring information from databases, difficulties may be experienced such as:

- Connected DB may not be able to provide answers to all the queries ES would like to put to it. The ES needs to distinguish between questions viable to put to a DB and that not.
- The ES and the DB may not be able to run in the same extensive memory needed for an ES and a DBMS on one machine, and
- Queries to the DB need to be phrased in a special DB query language. The ES should be able to formulate the query and interpret the reply.

The rules connect items of knowledge about a problem. Some ES concentrate more on facts and less on rules. Many pieces of information may have more than one attribute. Such knowledge is said to be structured, compound or multi-attributed. All ES languages should provide for it in some way.

- **Inferencing using production rules associated with objects**

The following describes the interaction between the knowledge base and the inference engine:
A production system typically consists of a rule base (knowledge base) and a rule interpreter (inference engine) that decides when to apply the rules to the data, goals and intermediate results in the working memory. The Working Memory (WM) holds the Object-Attribute Values (OAV) used to drive or fire the rules. The OAV triggers some rules in the WM by satisfying their conditions.

- **Inferencing using frames**

Non-monotonic reasoning expresses reasoning with default attributes designed for frames. The process retracts rules when proven wrong. Non-monotonic reasoning is a mathematical tool by which default information holds until this information becomes inapplicable.

♦ Blackboard subsystem

Blackboards are not an alternative to frames and nets for storing knowledge. It is rather a complex method used in complex systems to record choices, guesses and decisions about what to do next. It is used in situations where more than one source of information is active. All knowledge sources have access to a shared database. It involves an architecture that allows the independent knowledge sources to communicate through the central device: the blackboard.

When a knowledge source is activated, it examines the current state of the blackboard and implies its knowledge to either create a new hypothesis or change an existing one. The scheduler determines which knowledge source can contribute to the solution in the blackboard next. Hearsay-III - a speech-understanding project; ExperTax - a tax planning system, and FINSIM are examples of blackboard systems (Klein & Methlie 1995). Turban et al (2001) describes the blackboard as a kind of database that an area of working memory set aside for the description of the current problem, the input data and the intermediate results.

♦ **An explanation sub-system**

A by-product of the inference engine is that it is able to provide a trace of rules (subject knowledge) used and thus provide an **explanation** of its conclusions. This explanation can be presented to the ES user in the dialogue component. The explanation sub-system provides an area where the behaviour of the ES can be accounted for (Turban et al 2001). It provides answers to questions such as:

- How was a certain conclusion reached?
- Why was a certain question asked?
- What is the plan to reach the solution? , and
- Why was a certain alternative rejected?

♦ **A knowledge refining system**

Human experts can analyse their own performance, learn from it and improve it for future consultations. Having the program evaluate the reasons for success or failure and learning from it would greatly enhance ES. This functionality is not available in many commercial Expert Systems yet, but is being developed in experimental ES (Turban et al 2001).

6.2.3 The process of ES

The process of ES (See Figure 2-6: p21) can be divided into two parts (Turban et al 2001): the system **development** process in which the ES is constructed and the **consultation** process which describes how advice is rendered to the users. The **development** process starts with the knowledge engineer acquiring the knowledge from the expert. This acquired knowledge is then programmed in the knowledge base as facts about the subject and knowledge relationships in terms of if-then rules.

The **consultation** process involves the user who starts the process by requiring advice from the ES. The ES provides a conclusion and explanation, using its inference engine. The engine searches the knowledge base for an appropriate action by matching the input the user provides to the facts and rules in the knowledge base. To execute this task the inference engine uses a work area or temporary databases called the blackboard. All intermediate results are stored here. An explanation of actions taken by the inference engine can be provided to the user. Finally, the knowledge in the knowledge base may be refined by repetitive consultations. This knowledge refining system is not available in many Expert Systems now, but is being developed as experimental ES.

6.3 ES and expert shells

An expert shell is an ES without the domain-specific knowledge (Beynon-Davies 1991). Mallach (1994) refers to a shell as a pre-packaged inference engine. Construction time can be reduced by using an ES shell. Advantages of using a shell include (Beynon-Davies 1991):

- An expert shell can assist in rapid prototyping. The programmer needs only construct the knowledge component. The structure of the ES exists, enabling the developers to concentrate

on the substantive content rather than the form of the ES. A shell helps to reduce the level of skill required by the developers by effectively supplying some of the required expertise.

- A shell imposes prior structure, enabling developers to concentrate on substantive content rather than form, and
- A shell reduces the required skill level of Expert Systems builders

6.4 Knowledge acquisition and knowledge base construction

The knowledge that is contained in the system determines the effectiveness of the ES (See Paragraph 2.3.4: p22). Knowledge engineering may contain the following steps (See Paragraph 6.2.2: p113):

- Definition of the problem
- Acquisition of expert knowledge
- Design of an architecture, and
- Testing and refining of the program

Acquiring expert knowledge is a crucial component of knowledge engineering. This phase is difficult and time consuming. It is the process of gathering the relevant information about a domain, usually from an expert. Usually a computer program is compiled to incorporate all the knowledge of the domain. A number of knowledge acquisition techniques have been developed. As mentioned before knowledge can be classified as surface knowledge or deep knowledge. Surface knowledge involves the presentation of declarative and procedural knowledge into heuristics to quickly solve the problem. Surface knowledge is the basis for most ES and uses production rules. Deep knowledge involves fundamental knowledge of the domain including definitions, axioms, general laws, principles and causal relationships of the domain.

The human expert, from whom the knowledge is acquired, needs to be a person that can effectively cope with the problem domain. Selection of the person is based on reputation. The expert should be available and willing to co-operate in the process of developing the ES especially when acquiring the knowledge and testing the system. The expert must be able to communicate his expertise.

Validity of knowledge in any system is crucial. If the knowledge inferred proves to be faulty or erroneous in any way, the wrong decisions made by the experts will be made more speedily than when the process was a manual one. The sources of errors could include:

- Not much is known about the problem domain; too little knowledge is available. To solve the situation more domain expertise need to be obtained. , and/or
- The environment of the decision might be changing in which case the captured knowledge needs to be maintained

Relational database systems manage knowledge in the form of facts but are quite different when compared to knowledge bases. To understand ES and knowledge base construction, Beynon-Davies (1991) compares the two technologies.

6.4.1 Comparing a Knowledge-based System (or an ES) to a Relational Database System

Beynon-Davies (1991) states that knowledge can be represented as both facts and rules. Facts declare relationships between objects. Rules define the process by which new facts are generated from old facts. Rules are mechanisms to manage the information explosion in the attempt to represent reality. Beynon-Davies (1991) declares a database as a collection of structured data shareable between different parts of an organisation's information system, having properties such as program-data independence, data integration, data integrity and separate logical and physical views of the same data using a specific underlying data model. Hoffer et al (2002) calls these different models the database architecture. The different styles of database management characterised by the way data are defined and structured are called the database architecture. A database management system supports one of five architectures: Hierarchical, network, relational, object-orientated or multi-dimensional database models. Beynon-Davies (1991) compares knowledge-based systems to the relational model of databases when discussing the integration of expert and database technology. The relational model consists of tables representing the information stored on objects or entities and the relationships between them.

Important characteristics of relational databases are:

- *Data* are stored in a component called a *database*
- To use one data structure called a *table* or relation
- To handle large *collections of facts* representing declarative knowledge or relationships between objects
- Rules can be represented in databases only in a *declarative* manner, storing values in a separate table reporting the relationship. This may result in *many tables* to represent all the different relationships that exist between objects. To limit the number of tables some knowledge is *embedded* in the high-level language programs that accesses the database called the *front-end.*
- Rules as *procedural* knowledge are represented as constraints or additional tables mainly to govern *data integrity*
- A *high-level language* code interacting with the database contains and processes the procedural knowledge
- A database is a *limited Knowledge-based System*. It is a subset of what is normally meant by knowledge.
- A RDBMS (Relational Database Management System) is *program-data independent*. Data can be maintained independently of the programs that use it.
- Addition of a rule involves modifying the *structure* of the database (addition of a table structure) – not a simple matter
- *No inferencing* except when explicitly programmed
- Data must be valid. Rules govern the logical consistence of the database.

- A database uses a tool called the *database management system* (DMBS) to build database systems
- The DBMS (Database Management System) is built upon a *data model* or architecture, and
- The *database management system* (DBMS) is an integral part of the database and handles manipulation of large collections of facts

In contrast with DBMS, the characteristics of ES are:

- *Knowledge* (facts and rules) are kept as a separate component called a *knowledge base*
- The data structure depends on the knowledge representation chosen e.g. *production rules*, *structured objects* or *frames* and *predicate calculus* and others
- ES handle large *collections of rules* representing procedural or derivation of new facts from existing facts
- Rules as *declarative* knowledge are added independently from other rules or the process that uses it
- Rules as *procedural* knowledge are stored as data mainly used by the process to *derive new facts* from existing facts
- Knowledge can be represented as both *facts and rules*
- The knowledge base's *general-purpose processor* activates the procedural knowledge
- ES are *knowledge-process independent*. Rules are maintained independently of the process itself.
- Addition of a rule involves adding or deleting a *condition* – a relatively simple matter
- ES separate *inference* and knowledge. Once the inference process is in place, the knowledge worker can focus on the knowledge itself. ES is sometimes referred to as a knowledge-based system because knowledge is kept in a separate component as data.
- Knowledge may be incomplete
- ES use a tool called an *expert shell* to build Expert Systems
- An expert shell is built on some *formalism for knowledge presentation*, and
- The inference mechanism as an *integral part* of ES handles the activation of large collections of rules

A database system is an example of a limited Knowledge-based System (Beynon-Davies 1991). A knowledge base handles large amount of rules (procedural and declarative knowledge), while a database handles large amounts of facts or declarative knowledge. The procedural knowledge of a database is found in the associated external high-level language interacting with the database. The procedural knowledge (rules) are stored as data in a Knowledge-based System and activated by an inference process integral to the Knowledge-based System.

6.5 Summary

Expert Systems are based on two fundamental principles: the appropriate representation of domain knowledge and the control of the domain knowledge. A standalone expert system stores a few facts

and relies on the user to pass information to the system when needed to perform certain deductions. The combination of Expert Systems and database technology could benefit both technologies. Models have been developed for data base systems that would benefit the storage of knowledge in a knowledge base. DBMS technology could contribute Expert Systems by giving them the ability to access large collections of facts and also apply features such as concurrency control, data security and optimised access to knowledge base items (Beynon-Davies 1991).

A deductive component could enhance database technology by providing the rules component of an ES (Beynon-Davies 1991). A frame is roughly the equivalent to a row in a relational database. A slot is roughly the notion of an attribute in a relation. Frames and slots are artefacts used in ES rules to represent knowledge. A frame (See Paragraph 6.2.2: p113) use slots and is a knowledge presentation used in Knowledge-based Systems.

The writer concludes in agreeing with Turban (1995) that an ES component is ideal to assist a decision-maker in an area where expertise is required.