INTELLIGENT MONITORING AND CONTROL

Amílcar Cardoso
AI Lab, CISUC - Center for Informatics and Systems, University of Coimbra, Portugal

Abstract
Human operators of monitoring and control systems typically perform knowledge intensive tasks requiring human intervention in diverse and sometimes critical situations. The cognitive overload to which they are submitted may motivate human errors which compromise safety and performance. Expert Systems offer the means to codify and automate the use of the knowledge of the best experts in a given domain, and may provide support to monitoring, diagnosis and control tasks.
In this text we will make an overview of what expert systems are, what they may offer and what they don't. We will present their typical architecture and the most common knowledge representation formalisms and reasoning mechanisms, and discuss how they may deal with uncertainty. The life-cycle of an Expert System will also be presented, special attention being devoted to knowledge acquisition activities. An overview of some applications of expert systems to monitoring and control will be given, as well as some alternative approaches.

1. INTRODUCTION

The control needs of complex and expensive processes have motivated the development of sophisticated computational systems which allow the monitoring and control of complex processes by small crews of specialised human operators. Typically, these operators follow the process evolution through a set of instruments connected to a computer system which interacts with sensors, alarms and actuators. The control system is intended to minimize human intervention and to provide the operators with the maximum information on the controlled process. The operators' task essentially consists in trying to minimize failure risks while maximizing performance. The required degree of expertise depends on the complexity of the process and of the control system, on the importance, to people and equipment safety, of keeping the process in normal operation, and on the losses that an incorrect process control or its interruption may cause.

In normal steady-state conditions, the control system takes charge of keeping the system in a given set point, and the operators confine themselves to monitoring the most important variables. Human intervention is required during system start-up and shut-down, and also to regulate the process to given resources and needs.

Another, more critical, kind of situation occurs when equipment or process failures, or human errors, originate anomalous conditions. In such circumstances the control system presents a great quantity of information to the operators in a short time, originating in the most diverse sources, with the process conditions quickly changing and alarms escalating. In such a stressed situation, operators must discriminate actual faults from false alarm states, perform malfunction diagnosis, and plan and accomplish corrective actions[1]. This requires them to intensively resort to their knowledge on the process and on the control system. In particularly complex, stressed or time-critical situations, they

[1] According to [1], "monitoring activities track process variables intelligently and provide knowledge-based explanations of normal process behavior. (...) Fault detection (...) involves differentiating between normal and abnormal conditions. (...) Malfunction diagnosis isolates and identifies process malfunctions. (...) Corrective-action planning takes a diagnostic conclusion and provides a plan of action to deal with the problem safely and economically."

have difficulty in recognising the most relevant data, to evaluate the process state and to identify causes of the troubles. Consequently, they tend to perform inaccurate corrective actions and answer the problems in a wrong sequence. The cognitive overload they are submitted to withholds them to execute a correct, consistent and quick course of actions (see [2], [3] and [4] for actual examples).

Moreover, the more complex the process is, the most difficult becomes to the operators to understand its dynamic behaviour, particularly in unforeseen situations, and the most difficult becomes to guarantee a uniform level of expertise between different operators.

All these factors account for the need to endow control systems with tools aimed at giving on-line support to the operators' decision process, reducing the demanded cognitive load when abnormal conditions occur [2, 5].

Conventional programming techniques may be used to build such tools. However, they require quantitative models of the controlled processes, describing their whole range of operation, which usually is, in practice, an impossible task. Even when such models exist, they often demand the access to hard to obtain parameters [6]. Moreover, the resulting programs exhibit forbiddingly higher execution times with the increase of the process complexity [3].

Artificial Intelligence presents a set of favourable answers to these problems. Expert systems, in particular,

i) offer the means to codify the knowledge of the most expert operators and process designers, and

ii) to computationally automate its use, making it permanently available.

Their symbolic processing capability makes them a viable alternative to traditional monitoring and diagnosis methods, particularly when the controlled process is complex, difficult to model in a quantitative way, and the available knowledge is incomplete.

This paper intends to give an overview of how expert systems may contribute to improve monitoring, control and diagnosis. In the following section we will give an introductory picture of what expert systems are, what kind of applications they are intended to, and what kind of obstacles to their development are to be considered. In Section 3 the typical architecture of an expert system will be presented, as well as the main approaches to knowledge representation and reasoning. In Section 4 we will present some clues on how expert systems may deal with uncertainty. Section 5 will be devoted to the development of expert systems. In Section 6 we will return to monitoring and control, to analyse drawbacks of the expert systems approach, to point some emerging alternatives and to give an overview of the main applications of expert systems to those kinds of tasks. At last, in Section 7 we will draw some conclusions.

2. EXPERT SYSTEMS APPROACH

2.1 Characterisation of Expert Systems

Expert systems are knowledge-based computer programs, in the sense that they explicitly incorporate domain knowledge. They are intended to exhibit problem solving capabilities comparable to those of human experts. To attain a high level of expertise, they are restricted to solving problems in narrow knowledge domains, in contrast with some other knowledge-based programs which adopt a generalist approach. The knowledge they embody is acquired from human experts, and specific methodologies exist to guide this knowledge acquisition process. Once knowledge is incorporated, the program applies symbolic reasoning algorithms to infer new information from available knowledge and data.

Expert systems have the ability

i) to incorporate expert knowledge, including heuristics and rules-of-thumb,

ii) to deal with uncertain information and incomplete knowledge, and

iii) to explain the reasoning process underlying a founded solution.

They also offer easy integration of new knowledge and maintenance of the existing one. These characteristics make them especially useful when

i) there is an insufficient number of experts or they tend to disappear

ii) experts are needed in different places or in hostile environments

iii) experts are subjected to cognitive overload

2.2 Applications

Typically, applications of expert systems fall into two classes of problems: classification and construction.

In classification problems, a set of data is given and the expert system attempts to find which of a predetermined number of classes the data corresponds to. Medical diagnosis, equipment diagnosis, plant classification and process monitoring are examples of classification problems.

In construction problems, the expert system is given a goal and a set of constituent parts, and attempts to build an arrangement of the parts which may attain the goal. Examples are product design, computer configuration, route planning and production scheduling.

The majority of the most well-known expert systems deal with classification problems. Examples are, among the early ones, DENDRAL [7] (chemical identification), HEARSAY I and II [8] (speech recognition), INTERNIST/CADUCEUS [9] (internal medicine), MYCIN [10] (blood infections) and DELTA/CATS [11] (maintenance of diesel-electric locomotives). The classical example of an expert system dealing with a construction problem is R1/XCON [12] (VAX computers configuration at DEC).

Depending on the concrete problem to solve and on the environment where an expert system is expected to operate, we may consider several possible configurations for it: the expert system may work in a stand-alone fashion, interacting only with human users; it may be integrated in a computational environment, interacting with it and also with human users; it may be completely embedded in the computational environment.

2.3 Applications in Monitoring and Control

Expert systems for monitoring and control applications must satisfy some specific requirements [13, 14]:

i) they must dispose of efficient means to data acquisition;

ii) they must have the capability to deal with time varying data;

iii) the reliability of their answers must degrade in a graceful way when the available time to get the answers decreases;

iv) they must exhibit a robust behaviour: they must operate continuously while the process is running, without significant performance degradation when abnormal or unexpected conditions occur;

Although these requirements usually are not easy to attain, things get harder to manage when hard real-time constraints are present. Typically, in real-time applications many decisions are required in a timely fashion, facts have limited life spans, the process generates large amounts of data, operations are tied to clock time or time intervals, control decisions depend on current and historic data, and many asynchronous, concurrent operations may occur [15]. This makes the application of expert systems to real-time applications a current hot topic of research, despite the existence of several successful reported systems. We will come back to this topic later in Section 6.

2.4 The Knowledge Acquisition Bottleneck

One of the main obstacles to expert systems development lies in the knowledge acquisition task, aimed at eliciting knowledge from a human expert, and incorporating that knowledge in the expert system. This is a very hard and time consuming task, due to a series of reasons:

i) when the development starts, the expert system developer usually doesn't know anything at all of the expert's domain of knowledge, and must learn enough about the expert's problem solving task to build the system; this is basically done through a series of interviews with the expert, which must be carefully planned to ensure successfulness;

ii) usually experts have great difficulties to understand what an expert system really is;

iii) in consequence, when describing to the expert system developer how they perform their activity, experts risk to emphasise irrelevant information and to neglect important one;

iv) sometimes, experts are not motivated to collaborate with the developers, or they are too busy to devote enough time to the interviews.

Given the importance of knowledge acquisition to the success of an expert system project, researchers and field developers have proposed several methodologies to guide not only this activity, but also the development process itself, as both are tightly linked. Considering the introductory character of this course of lectures, in this paper we will devote Section 5 to this topic, adopting pragmatic guidelines originating in field developers. We must stress, however, that more formal methodologies have been proposed, the most important of all is KADS [16, 17], which results from an ESPRIT funded joint effort.

3. EXPERT SYSTEMS ARCHITECTURE[2]

An expert system is a program whose degree of expertise is intended to be comparable to that of a human expert in a specific domain of knowledge. To attain this purpose, they rely on the incorporation of expert knowledge which the program uses for problem solving.

Expert knowledge has some specificities which are worth mentioning:

i) it is seldom based on clear definitions or on precise algorithms;

ii) it is composed of general theories, but also of heuristics, which are used to simplify problem solving, to identify common situations, and so on;

iii) it is continuously changing.

These specificities influenced the architecture of expert systems. Actually, a uniqueness of expert systems lies in the clear separation between knowledge, on one hand, and algorithms to apply knowledge, on the other hand. This separation results in the existence of two main modules (see Figure 1): the Knowledge Base (KB) and the Inference Engine (IE). To complete the basic architecture, a User Interface (UI) guarantees communication with the outside world.

This architecture greatly favours expert systems development and maintenance, because we may freely change the KB contents (the evolving part of the system) without having to re-program the remainder of the system. Also, we may develop efficient algorithms when building the IE, without caring with the specificities of a particular domain.

[2] This section is based on [18] and [19].

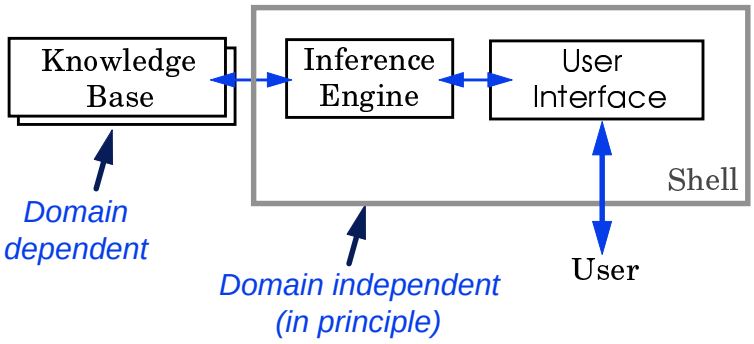

Figure 1 - Basic Expert System Architecture

The algorithms of the IE are in principle domain independent. Also, the UI may usually be composed by generic mechanisms. The KB is obviously domain dependent and actually determines the characteristic behaviour of a specific expert system. Roughly, given an expert system which solves problems in a domain A using the Knowledge Base KBA, if we substitute KBA with a Knowledge Base KBB describing domain B we will get an expert system which solves problems in domain B. This is the principle behind the idea of joining the IE/UI modules in one domain independent tool, called Shell. There are many commercially available Shells which may be used to build expert systems. With such a tool, the expert system builder may concentrate on the KB construction, and use the generic inference and interface mechanisms offered by the Shell, instead of programming the system from scratch.

3.1 Knowledge Base and Representation Formalisms

A Knowledge Base contains domain knowledge, which may be represented through different formalisms. Knowledge representation formalisms must simultaneously be clear and readable for the ES developer, be computationally workable, exhibit adequate expressive power, and be easy to update and edit. We are going to present some of the most common alternatives.

3.1.1 Object-Attribute-Value Triplets

One of the simplest formalisms is the OAV Triplet (see Figure 2), which may conveniently represent facts in the Knowledge Base. Given a relevant object of the domain (e.g., an aeroplane), we may define a set of attributes which characterises the object (e.g., the number of engines, the engine type and the wing design). Each attribute may have one or more values (e.g., number of engines = 4). Figure 3 illustrates how we may use OAV triplets to represent three facts about a specific aeroplane.

A common simplification of this formalism is the Attribute-Value Pair, which makes the object implicit. When using this representation, it is advisable to include some reference to the implicit object in the attribute name (e.g., “Number of Engines of C130 = 4”). Often this inclusion is mandatory because attributes in AV pairs must have unique names (this is not the case in OAV triplets, where different objects may have attributes with the same name).

Figure 2 - OAV Triplet

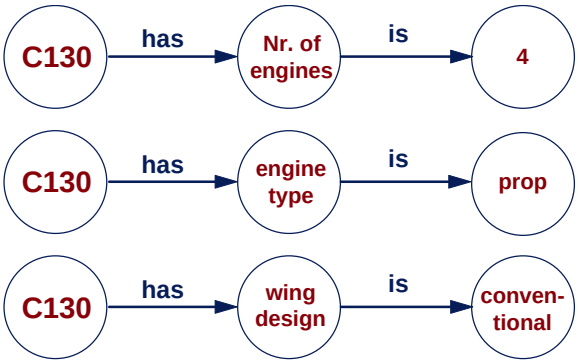

Figure 3 - Three OAV Triplets

3.1.2 Semantic Networks

We may consider Semantic Networks as a generalisation of OAV Triplets. A Semantic Net is a directed graph with labelled edges and nodes (see Figure 4). While an OAV triplet represents the relation between *one* object, *one* attribute and *one* value, a semantic net may represent *several* objects, *several* attributes per object, and even *relations* between objects. Relationships are of prime importance to organise and use knowledge. With the ability to relate objects, we may increase the expressiveness and versatility of the knowledge base. For instance, we may create nodes representing classes of objects sharing common characteristics, and link the nodes with sub-class/super-class relationships.

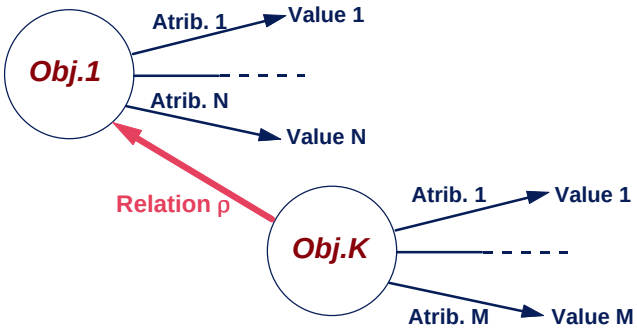

Figure 4 - Semantic Network

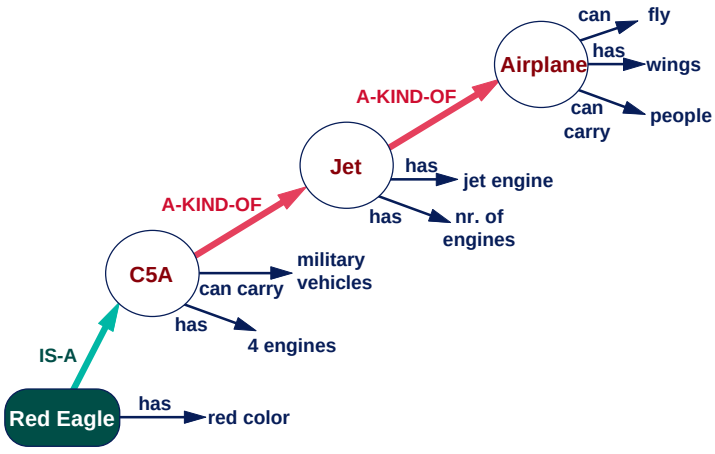

Figure 5 - Partial view of a Semantic Network

The semantic net of Figure 5 illustrates the use of two very useful relations: IS-A, which relates a class and a specific instance of that class, and A-KIND-OF, used to express sub-class relationships. These relations are used to represent inheritance between objects, which may significantly enhance expressiveness and reduce memory storage requirements.

3.1.3 Frames

Frames have similarities with semantic nets in that they offer the means to represent objects, their attributes and values, and relationships between them. Roughly speaking, they are sophisticated data structures composed of a set of non-homogeneous *slots* which may be *filled* with information, each slot corresponding to an object attribute. In contrast to the simple object representation that we have seen up to now, instead of storing just values, slots may contain relations, default values, pointers, rules and even procedures (see Figure 6). Frames may also be used to build semantic networks, offering a very rich representation for their nodes. Actually, we may think of a frame as an extension to the O-A-V representation.

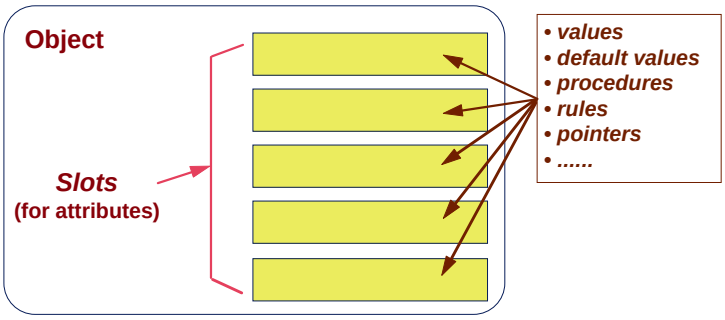

Figure 6 - A Frame

Figure 7 illustrates the use of frames to represent the concept of *dog*. Relations, default values, ranges of values and procedures may be observed in this frame.

Frames are a powerful, robust, versatile and expressive knowledge representation formalism, but they are difficult to use and their behaviour is hard to predict. However, they have proved effective for large-scale, complex systems, involving default values and a large amount of a priori facts.

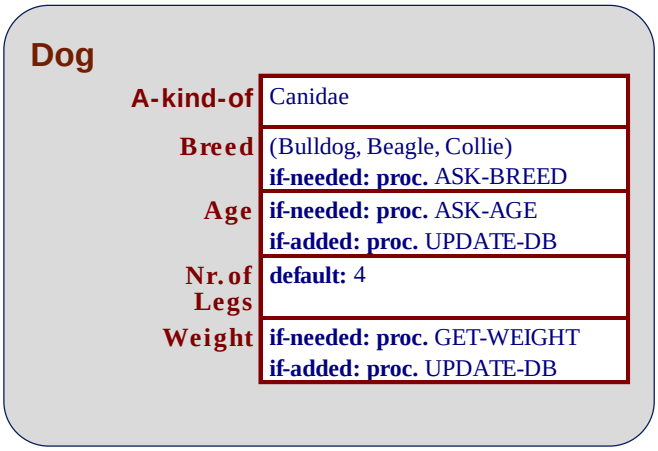

Figure 7 - Example of a Frame

3.1.4 Predicate Logic

Predicate logic is an extension of propositional logic which may be used to represent the components of a sentence. Its simplest form, the first order predicate logic, may represent rather complex sentences. For instance (example from [18]), we may write

- mammal(dog), meaning "a dog is a mammal", which is a true sentence;
- four_legs(dog), meaning "a dog has four legs", which is usually true;
- mammal(chicken), meaning "a chicken is a mammal", which is false;
- sister(joan, jack), meaning "Joan is Jack's sister"; we don't know if this is true or not because it depends on who actually are Joan and Jack;
- sister_of(steve, mary) AND cousin(mary, ann), meaning "Mary is Steve's sister and Ann's cousin";
- ill-humoured(boss) AND ask_for(boss, rise) ⇒ bad(idea), meaning "Asking for a rise when the boss is ill-humoured, is a bad idea";
- human(X) ⇒ mortal(X), meaning "All humans are mortal"

Predicate logic has the advantage of offering a well established syntax and a clear semantics for knowledge representation, and a founded method, called resolution, for inferring new information. However, implementing resolution is not a trivial task, so the use of predicate logic in commercial shells is not common. Even so, there are programming languages (e.g., Prolog [20]) which implement predicate logic allowing the development of expert systems using this knowledge representation formalism.

3.1.5 Production Rules

Production rules, usually called just Rules for the sake of simplicity, are the most used knowledge representation formalism in Expert Systems. They origin from theories of cognitive psychology which proposed production rules as models for knowledge storage in *long-term memory*. According to those theories, sensory input provides stimuli to the brain, triggering rules of long-term memory, resulting in the production of responses [21]. Rules are intended to model small chunks of knowledge, providing modularity to knowledge representation.

The most general form of production rules is:

Name: IF Antecedent THEN ConsequentA ELSE ConsequentB (1)

where *Name* is a rule identifier, *Antecedent* has the form

Condition1 AND Condition2 AND ... AND ConditionN (2)

and ConsequentA, ConsequentB have the form

Conclusion1 AND Conclusion2 AND ... AND ConclusionM (3)

The meaning of a rule with this form is: if *Antecedent* is true, then conclude that *ConsequentA* is true, otherwise conclude that *ConsequentB* is true.

Usually rules assume the simpler IF-THEN form. Actually, some shells only support this form. Fortunately, an IF-THEN-ELSE rule may be substituted by two IF-THEN corresponding rules. For instance, the rule

Rule 1: IF sensor A has value greater than 50

THEN water temperature is too high

ELSE water temperature is normal

may be substituted by the rules

Rule 1a: IF sensor A has value greater than 50

THEN water temperature is too high

Rule 1b: IF sensor A has value lesser than or equal 50

THEN water temperature is normal

The Antecedent of a rule may also include the disjunctive operator OR. Experience says that the use of this operator reduces readability and makes debugging harder, but it is possible to substitute a rule with a disjunctive antecedent by two rules with conjunctive antecedents. For instance, the rule

Rule 2: IF sensor A has value greater than 50

OR sensor B has value greater than 45

THEN water temperature is too high

may be substituted by the rules

Rule 2a: IF sensor A has value greater than 50

THEN water temperature is too high

Rule 2b: IF sensor B has value greater than 45

THEN water temperature is too high

It is worth noting that we may consider a rule as relating several Object-Attribute-Value triplets. Figure 8 illustrates this for Rule 1a above.

	IF	*THEN*
Object	***Sensor A***	***Water***
Attribute	***has value***	***temperature***
Value	***• 50***	***too high***

Figure 8 - A rule as a relation between OAV triplets

3.2 Inference Engine and Inference Mechanisms

In an expert system, the algorithms to process knowledge are grouped in the Inference Engine module. Its job is to process the contents of the KB, and the information coming from the UI, to infer new information. Therefore, the algorithms in the IE must use inference mechanisms capable of dealing with the knowledge representation formalisms used in the KB. For instance, when using Predicate Logic to represent knowledge, we may use Resolution to infer new information. When using Semantic Networks and/or Frames, we typically recur to Inheritance to process A-KIND-OF and IS-A relations. As Production Rules are the most used representation formalism, in this text we are going to focus on inference mechanisms to deal with them.

The basic mechanism to process rules is Rule Chaining, which consists in connecting a problem to its solution through a chain of rules. To produce the chain, the antecedent conditions of each rule are linked to matching conclusions of other rules. Inference is performed by transversing the chain. Depending on the direction of the transversal, two basic chaining methods result: Forward Chaining, from the problem to the solution, and Backward Chaining, in the opposite direction. For the sake of simplicity, in the remainder of this section we will restrict our analysis to IF-THEN rules,

with conjunctive antecedents (no OR operator) and a single conclusion. As we have already seen, this is not actually a restriction, since rules not conforming these constraints may be converted to the simpler form that we are considering.

3.2.1 Forward Chaining

Forward chaining is fact-oriented: we start with a set of Facts, and use the rules to deduce new facts. The basic algorithm is:

i) Given a set of Facts, chose a rule whose Antecedent is true, and fire the rule; the Consequent of the rule is a new Fact

ii) Use the new set of Facts to (recursively) proceed

iii) Stop when no more rules may be fired

Figure 9 illustrates the method. The initial set of facts consists of C, D and E. With this set, rule R2 is triggered, producing the new fact G (its conclusion). With the expanded set of facts, rule R5 is triggered, and the new fact H is added to the set of facts. As no more rules may be fired with the current set, the process is stopped.

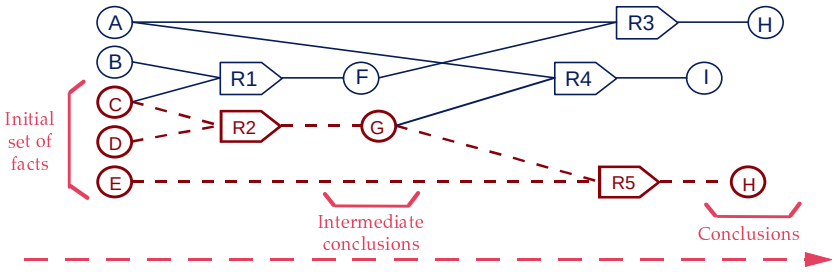

Figure 9 - Forward Chaining

3.2.2 Backward Chaining

Contrasting to the former mechanism, Backward Chaining is Goal-oriented: given a hypothetical solution (a Goal), we use the rules to seek for support to that solution. A rule supports a goal when its conclusion matches it and its antecedent is true. Therefore, seeking for goal support results in searching backwards through a chain of rules. The basic algorithm is:

i) Given a Goal, stop if a Fact supports it

ii) Otherwise, chose a rule whose Consequent may produce that Goal

iii) Consider the Conditions of the Antecedent as new Goals, and (recursively) proceed

Actual shells adopt a refined version of this algorithm which assumes that the set of available Facts may be empty when the inference process starts. As a consequence, if no rule may be chosen in the second step, the algorithm asks the user about the truthfulness or the falseness of the Goal, giving rise to the characteristic dialogues of most expert systems.

The example of Figure 10 illustrates the mechanism. Given the hypothesis H, two rules may support it: R3 and R5. Let's suppose that the algorithm chooses R3 according to some criteria. To be triggered, R3 must have a true antecedent, therefore its conditions A and F are new Goals to attain. As there is no rule which may support A, the user is questioned about its value. If the user states that it is true, A is asserted as a new Fact. As regards to F, the rule R1 is selected, and B and C are new goals. The user is questioned about their values, and if the answer is positive they are asserted as new Facts. Now, the Facts B and C may be used to fire rule R1, giving support to the goal F, which is asserted as a new Fact. From F and A, the rule R3 may be triggered, which confirms the initial Goal H as a Fact.

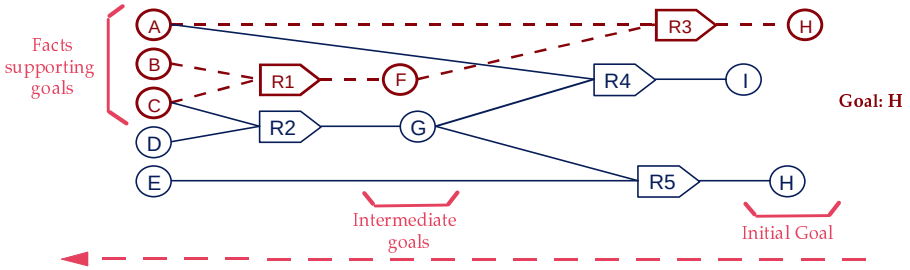

Figure 10 - Backward Chaining

3.2.3 Forward vs. Backward Chaining

Forward chaining, being data-driven, tends to be more effective than backward chaining when the set of possible conditions is small, while the latter suits better to tasks with a limited set of possible conclusions.

In addition to this computational characterisation, the adoption of one of the mechanisms tends to lie on the specific characteristics of the task to perform. For instance, some tasks like monitoring are mainly data-driven (we get the data from the process, then we try to understand which process state conforms to the data), suggesting the use of forward chaining. Conversely, diagnosis tasks are typically goal driven (we hypothesise a cause, afterwards we seek for data which confirms/contradicts the hypothesis), which conforms to the fact that the great majority of expert systems for diagnosis use backward chaining. This should be understood just as a general guidance to the selection of the best mechanism to choose. It doesn't mean that we should always adopt forward chaining when dealing with data-driven tasks, and backward chaining in goal-driven ones.

Careful analysis of real tasks show that the resort to both kinds of mechanisms, in a more or less intermixed way, may improve performance and quality of results. This has led to the adoption of mixed modes of chaining to opportunistically improve the performance of expert systems. The idea is to alternatively use both mechanisms during a problem solving session, according to a task dependent strategy. A common strategy gives priority to backward chaining, restricting the use of forward chaining to situations when it is not possible to backward chain. The converse strategy, which gives priority to forward chaining, is also commonly used.

A potentially better strategy consists in choosing one mode of chaining according to the current phase of the problem solving process. For instance, consider Monitoring and Control. Given a set of data, collected from the process, we may use forward chaining to reach to a limited set of candidate process states conforming to the available data. These candidate states may be seen as hypotheses which require further discrimination. This may be accomplished with backward chaining, which guides the acquisition of additional data (from the process, from the user, from historical data, ...) to confirm/contradict each hypothesis, and (hopefully) select only one. Given the determined process state and the overall data collected from the process, we may use forward chaining to determine the best control action to take.

3.2.4 Generating Explanations

The separate representation of knowledge in expert systems gives them the capability to easily generate explanations. The most common ones are produced as answers to "How?" and "Why?" queries. In rule-based systems, explanations are generated from the analysis of the rule chaining.

Consider the chaining in Figure 10. After reaching the conclusion H, the expert system may answer the user question "How did you choose H?" by tracing the fired rules. A possible answer is represented in Figure 11. This kind of explanation is particularly important to increase the confidence of the human user in the system, and also for debugging purposes.

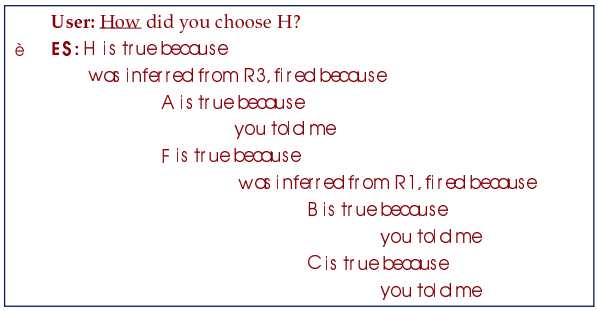

Figure 11 - "How" explanation for the chaining of Figure 10.

The expert system may also answer "Why?" questions during the rule-chaining process, as in Figure 12. In this example, the expert system had just questioned the user about the truthfulness of B, and the user decided to query the system about the reasons for its question. With this kind of explanation the human user may understand the context of a specific system question.

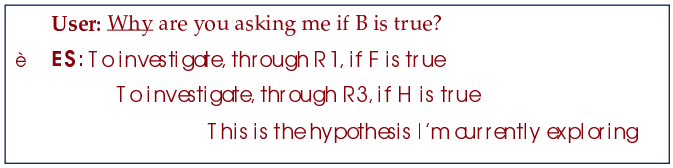

Figure 12 - Example of "Why?" explanation for the chaining of Figure 10.

4. DEALING WITH UNCERTAINTY

We may think of a rule "if E then H" as stating that the antecedent E is an evidence supporting the hypothesis H. Up to now, we have considered that E may assume the values "true" or "false", i.e., we may establish its truthful value with absolute certainty by some means, e.g., through inference or data gathering. Also, we have considered that the evidence E would confirm hypothesis H with absolute certainty.

In many real world domains things are not that easy, and expert systems must deal with uncertainty originating in data and knowledge. Often determining E with absolute certainty is not possible or easy to attain: e.g. data collecting equipment may be inexact or inaccurate, getting all the needed data may require expensive or dangerous tests. This requires the association of a measure of uncertainty with each peace of information.

Moreover, most of the rules in a KB may represent heuristics which the expert uses to reach conclusions without complete information, i.e., with only partial evidence. A rule of this kind represents the fact that an evidence E only partially supports a hypothesis H. To deal with this, we must associate some measure of uncertainty to the rule, representing the degree of support of evidence E on hypothesis H.

Besides some form of uncertainty quantification in rules and data, a method for combining uncertainty values is also needed. Some of the approaches adopted in expert systems to deal with uncertainty are:

- Classical Probability Theory
- Bayesian Probability
- MYCIN Certainty Factors [22]
- Fuzzy Theory [23]
- Dempster-Shafer Theory of Evidence [24]

The main distinction between the different methods lies in the way how each one conceptualises and manipulate uncertainty. In this text we are going to focus on two of the most used formalisms: MYCIN Certainty Factors and Fuzzy Theory.

4.1 MYCIN Certainty Factors

In probabilistic approaches, uncertainty is viewed as the probability of a peace of information being true[3]. MYCIN developers adopted a different view which proved to be effective in medical applications and was also applied in many other domains: they conceptualised uncertainty as a degree of belief in a fact or rule[4].

Let us denote cf(A) as the certainty factor of assertion A, with values ranging from -1 to 1, *cf*(A) = 1 meaning that there is absolute certainty that A is true, *cf*(A) = -1 meaning that there is absolute certainty that A is false, and *cf*(A) = 0 meaning that the truth value of A is unknown.

A MYCIN-like rule has the form

$$\text{Name: IF Evidence THEN Hypothesis } (cf = CF_R) \tag{4}$$

where CF_R, the rule's certainty factor, denotes the belief in Hypothesis given that Evidence is observed. A rule will be fired when cf(Evidence) ≠ 0. When calculating *cf*(Hypothesis), the CF of the rule serves as an *attenuation factor* for the certainty of the rule's antecedent:

$$cf(\text{Hypothesis}) = CF_R * cf(\text{Evidence}) \tag{5}$$

When the antecedent of a rule is a conjunction, the evidences of the conditions must be combined. The computation follows the principle that "the strength of a chain is determined by its weakest link". A *threshold level* δ is also used to prevent firing rules producing Cfs close to zero, which would result in a propagation of near-zero confidence values, and therefore poorly useful information, which consequently would give rise to a loss of performance. In MYCIN, a threshold level of 0.2 was used.

Given $E = C_1 \wedge ... \wedge C_n$,

$$\text{if } cf(C_i) \geq \delta \text{ for all } i,\ cf(E) = \min [cf(C_1), ..., cf(C_n)] \tag{6}$$

$$\text{if } cf(C_i) \leq -\delta \text{ for all } i,\ cf(E) = \max [cf(C_1), ..., cf(C_n)] \tag{7}$$

$$\text{if } |cf(C_i)| < \delta \text{ for any } i,\ cf(E) = 0 \tag{8}$$

$$\text{if any two } cf(C_i) \text{ are of opposite sign, } cf(E) = 0 \tag{9}$$

Evidences must also be combined when two fired rules have the same conclusion (consonant rules). The computation is performed using a set of heuristic formulas.

Given two rules with the same conclusion *H*, let *cf1* and *cf2* be the certainty factors computed for *H* with each one of the rules. The resulting CF for *H* is:

$$\text{if } cf1 > 0 \text{ and } cf2 > 0,\ cf(H) = cf1 + cf2 - cf1 * cf2 \tag{10}$$

$$\text{if } cf1 < 0 \text{ and } cf2 < 0,\ cf(H) = cf1 + cf2 + cf1 * cf2 \tag{11}$$

$$\text{if } -1 < cf1 * cf2 < 0,\ cf(H) = (cf1 + cf2) / [1 - \min (|cf1|, |cf2|)] \tag{12}$$

---

[3] For a detailed analysis of probabilistic approaches, their advantages and drawbacks, see for instance [21], Chapter 4.

[4] Actually, in MYCIN the certainty factor was a composite of a measure of belief MB and a measure of disbelief MD. In this paper, for the sake of conciseness, we adopt the simplified account of the methodology described in [18].

$$\text{if } cf1 * cf2 = -1,\ cf(H) = 0 \tag{13}$$

These formulas may be generalised for any number of consonant rules by applying them to two rules at a time. Note that, by (10) and (11), when multiple rules jointly confirm (disconfirm) a same hypothesis, its confidence asymptotically approaches 1 (-1). Also, if a single rule produces a confidence of 1, the combined confidence will be 1, except if a contradictory result of -1 is produced by another rule.

4.1.1 Example

Consider the next set of rules:

R1: IF A=i AND B=j THEN X=x (*cf* = 0.8)

R2: IF A=k AND B=l THEN X=y (*cf* = 0.8)

R3: IF C=m AND D=n THEN X=x (*cf* = 0.9)

R4: IF C=o AND D=p THEN X=y (*cf* = 0.4)

Suppose that the following certainty factors are known:

cf(A=i) = 0.6 cf(A=k) = 0.4 *cf*(C=m) = 0.4 *cf*(C=o) = - 0.3
cf(B=j) = 0.5 *cf*(B=l) = 0.9 *cf*(D=n) = 0.5 *cf*(D=p) = - 0.8

Applying formulas (5), (6) and (7), we get the CFs computed from each rule:

$cf(X=x)_{R1}$ = min (0.6, 0.5) * 0.8 = 0.5 * 0.8 = 0.4

$cf(X=y)_{R2}$ = min (0.4, 0.9) * 0.8 = 0.4 * 0.8 = 0.32

$cf(X=x)_{R3}$ = min (0.4, 0.5) * 0.9 = 0.4 * 0.9 = 0.36

$cf(X=y)_{R4}$ = max (- 0.3, - 0.8) * 0.4 = - 0.3 * 0.4 = - 0.12

From these results and formulas (10) and (12), we get:

cf(X=x) = 0.4 + 0.36 - 0.4 * 0.36 = 0.76 - 0.14 = 0.62

cf(X=y) = (0.32 - 0.12) / [1 - min (0.32, 0.12)] = 0.2 / 0.88 = 0.28

We may conclude that the value for the attribute X with a greater degree of belief is “x”.

4.2 Fuzzy Theory

Fuzzy Theory is easy to understand in terms of Set Theory. Classical sets may be discrete, when members take values from a discrete set (e.g., the set of models of cars with ABS made by a given manufacturer), or continuous, when members take values from a continuous set (e.g., the set of positive reals above 10). We may represent a continuous set through a *membership function* (Figure 13[5]) which assumes the value 1 for members of the set and the value 0 for non-members.

Suppose we want to represent the set of *small numbers*. If we assume that a number is small if it is lesser than a given number, let’s say 6, we obtain the membership function of Figure 14.

[5] The figures and the example in this section were adapted from [25].

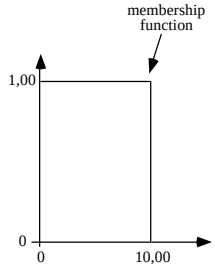

Figure 13 - Membership function for the set of positive reals above 10

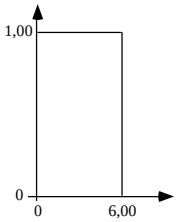

Figure 14 - Membership function for the set of small numbers

This representation clearly has a problem: even if we consider that the number 6 is a good reference to distinguish between small and non-small numbers (which is arguable and depends on the context where we are working), how can we accept that 5.99 is a small number and 6.01 is not? The discontinuity we observe in the function is obviously not appropriate to represent fuzzy concepts like “small”, “tall”, “too hot” or “has fever”. The point is: membership to the set of small numbers must be defined in a continuum, each number belonging to the set *in a certain degree*, or *with a certain likelihood*. The membership function for “small numbers” in Figure 15 is clearly a better representation for the set.

In fuzzy sets, membership is defined by a continuous function with values ranging from 0 to 1. From this point of view, they may be considered as a generalisation of conventional sets. They are used to represent concepts whose definition is uncertain, and offer the possibility of using linguistic variables and qualifiers which commonly appear in expert rules.

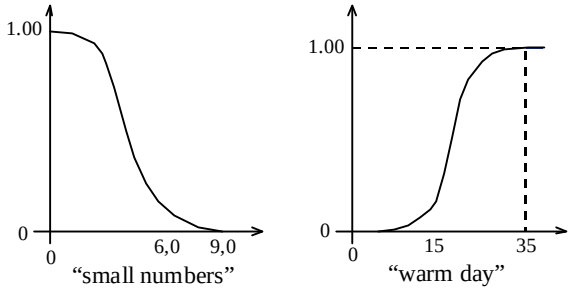

Figure 15 - Membership functions for fuzzy sets

Similarly to what happens with conventional logic, fuzzy logical operations may be performed by means of set operations (Figure 16): logical conjunction corresponds to set intersection, and disjunction to set reunion; negation of a membership function is computed by its complement to 1.

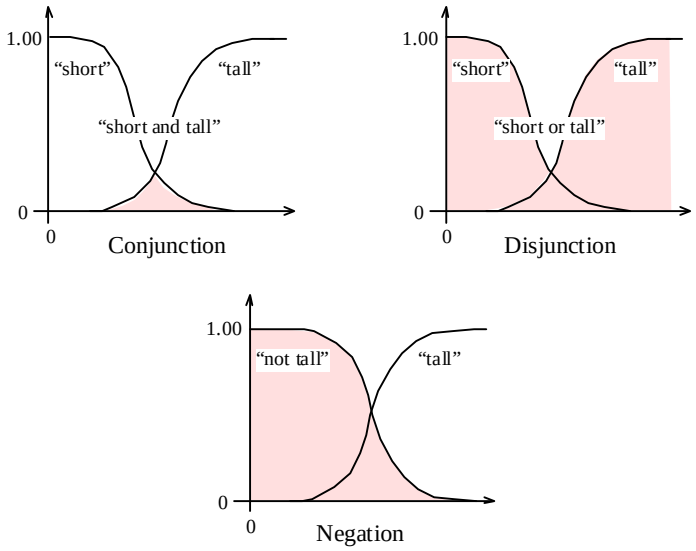

Figure 16 - Fuzzy logical operations

We also may use fuzzy rules, which represent the mapping between the membership functions of the antecedent and consequent. While in conventional rules we may infer that the consequent is true when the antecedent is also true, because the membership functions have just the values 1 or 0, in fuzzy rules the likelihood of the consequent depends on the degree of membership of the antecedent value to the antecedent fuzzy set.

Figure 17 illustrates how inference is performed. In this example, the sets for "the weather is warm" and "the beach is full" are graphically represented. If the temperature is known to be 27.5°C, we may say that the degree of membership of this temperature to "the weather is warm" is 0.3. Using the rule "IF the weather is warm THEN the beach is full", we may infer that the likelihood of "the beach is full" by mapping the antecedent degree of membership to the consequent set, which gives a result of 0.4.

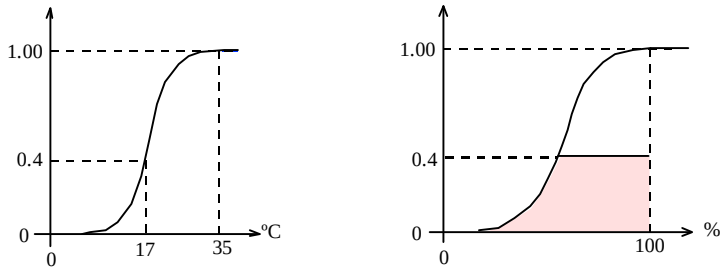

Figure 17 - Applying the fuzzy rule "IF the weather is warm THEN the beach is full"

The next example illustrates the use of fuzzy sets in a simple control application. We will use the example to introduce other important concepts of the fuzzy theory.

4.2.1 Example

Suppose we need to control the water temperature of a home shower by adjusting the mixing faucet. The values for the state variable "water temperature" are percentages of the whole range of temperature, and the values for the control variable "faucet opening" range from -2.0 (maximum cold water) to +2.0 (maximum hot water).

First of all, we must decide the vocabulary (qualifiers) to use for the two variables involved:

Table 1
Qualifiers for the fuzzy variables

Values for water temperature	Values for the faucet opening
freezing	much more cold water
cold	more cold water
warm	don’t move
hot	more hot water
scalding	much more hot water

Next, we must define the fuzzy sets for the qualifiers. The membership functions for the qualifiers of “water temperature” are represented in Figure 18, and those for “faucet opening” in Figure 19.

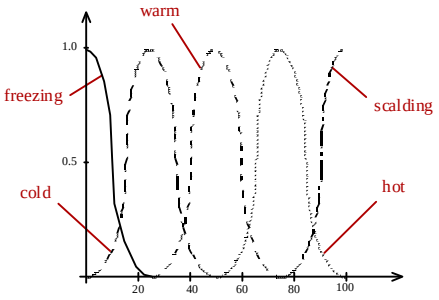

Figure 18 - Fuzzy sets for “water temperature” qualifiers

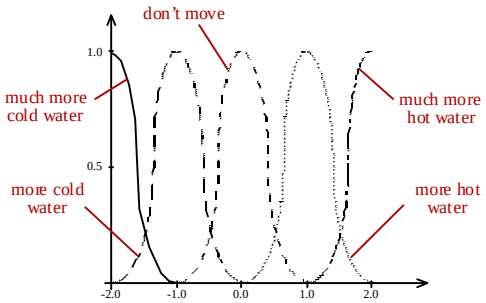

Figure 19 - Fuzzy sets for “faucet opening” qualifiers

Now we may define a set of fuzzy rules:

R1: IF “water temperature” = “freezing”
THEN “Faucet opening” = “much more hot water”

R2: IF “water temperature” = “cold”
THEN “Faucet opening” = “more hot water”

R3: IF “water temperature” = “warm”
THEN “Faucet opening” = “don't move”

R4: IF “water temperature” = “hot”
THEN “Faucet opening” = “more cold water”

R5: IF “water temperature” = “scalding”
THEN “Faucet opening” = “much more cold water”

Suppose that the user classifies the water temperature as “almost good” and we are able to map by some means this qualifier to 40% of the temperature range (see Figure 20). Alternatively, you may suppose that the input value was directly measured.

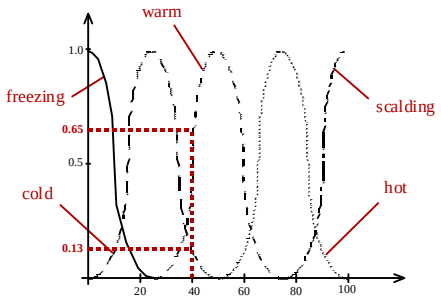

Figure 20 - Membership degrees for “almost good”

As we may see in the figure, the degree of membership of this temperature is 0.68 for the set “good”, 0.32 for the set “cold” and zero for the remainder. The process of converting a real number to a set of degrees of membership to fuzzy sets is called *Fuzzyfication*. From the rules R2 and R3, we may map these values to the “more hot water” and “don’t move” fuzzy sets. The resulting consequent’s likelihood is represented in Figure 21.

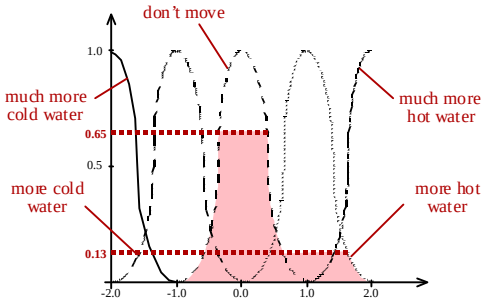

Figure 21 - Consequent’s likelihood

Now the problem is confined to how to decide the control action to take: we must convert the output of the inference in a non-fuzzy (crisp) value, i.e., a real number. This is called *Defuzzification*, and there are several methods to do it. One possibility is to choose the action with maximum likelihood. In this case (see Figure 22) there is a maximizing interval, therefore we chose the median of the maximizing set, which is 0. This is called *Mean-of-Maximum Method*, or simply *Maximum Method*.

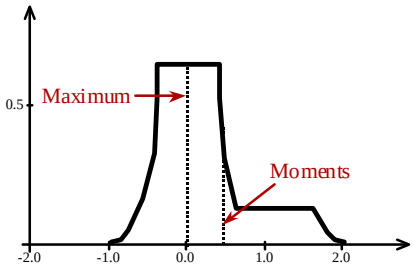

Figure 22 - Two defuzzification methods

An alternative is the *Moments* (or *Center-of-Area*) *Method*, which consists in computing the center of the area of the likelihood function. In this case (see Figure 22), the result is 0.3, which corresponds to slightly opening the hot water[6].

[6] For a more detailed view of defuzzification methods, including others not mentioned in this text, see [26], Section 6.3.

The Maximum Method is particularly suitable for applications which require a decision from a set of discrete values (e.g., "buy gold" vs. "buy stocks"). Note that with the rules and sets of this example, if we make the input value change continuously over its range the output value will exhibit discontinuities and will assume values over a discrete range when we use this method. Conversely, with the Moments Method the output behaviour will change continuously, which makes the method more appropriate for control applications. This method has, however, some drawbacks: besides being computationally expensive, there are situations where it produces unsuitable results, because the center of the area may be a value with, let's say, zero likelihood.

5. DEVELOPING EXPERT SYSTEMS

5.1 The Life-Cycle of an Expert System

Knowledge Engineering is the branch of AI which deals with the expert systems development process. Among the activities of a Knowledge Engineer we may mention: knowledge acquisition; assessment of the suitability of knowledge-engineering technology for a prospective application; planning and management of an ES project; design of appropriate representation formalisms; design of inference and control mechanisms; implementation or selection of a suitable ES shell; design and implementation of a KB; integration of an ES with its operating environment; evaluation of an ES. The most difficult and tricky of them is Knowledge Acquisition (KA), for the reasons we have already pointed out in Section 2.4. Moreover, the other activities have identical counterparts in general Software Engineering, contrarily to KA, which is characteristic of knowledge-based systems.

Many methodologies have been proposed to guide KA, and consensus is far from being reached among researchers and practitioners in the area. In this text we present an approach[7] which we have applied with satisfactory results in the development of many small-to-medium size ES.

Under this approach, the life-cycle of an expert system is represented in Figure 23. There are many similarities with the life-cycle of other software products: we find the same four phases in many Software Engineering methodologies, and the same happens with many of the Steps. Nevertheless, the obstacles associated to KA have led to the adoption of two development techniques which, although being used in general software engineering, assume a key role in knowledge engineering: cyclic development and early prototyping.

As we may see in Figure 23, cyclic development may actually embrace the development and deployment phases, and involves two types of cycles: Extension Cycles, intended to broad the scope[8] and capabilities of the ES, and Refinement Cycles, intended to increase the knowledge accuracy. With this approach the knowledge engineer may focus his KA activities in small amounts of knowledge in each cycle, which facilitates his communication with the expert and enables a solid deepening of the acquired knowledge.

The first cycle is obviously of the first type, and produces an initial working prototype. Although limited in scope, this prototype usually represents a key factor in the success of the ES overall development: first, because it may determine the enthusiasm of the expert, which often discovers that "it works" when faced with the prototype; second, because frequently the financial provision for the overall project is assured only when a working prototype is shown to the potential supporters.

In subsequent iterations, the prototype is gradually refined or extended in scope and capabilities. This evolving prototype constitutes a very important KA support tool, as it favours

[7] In this Section we follow the methodology presented in [27]. Also, the Figures of the Section are reproduced from this book.

[8] The scope of the ES is viewed as the extent of the task that it may perform and the range of situations in which it can perform the task [27].

communication with the expert and may function as a testbed for the validity and completeness of the acquired knowledge.

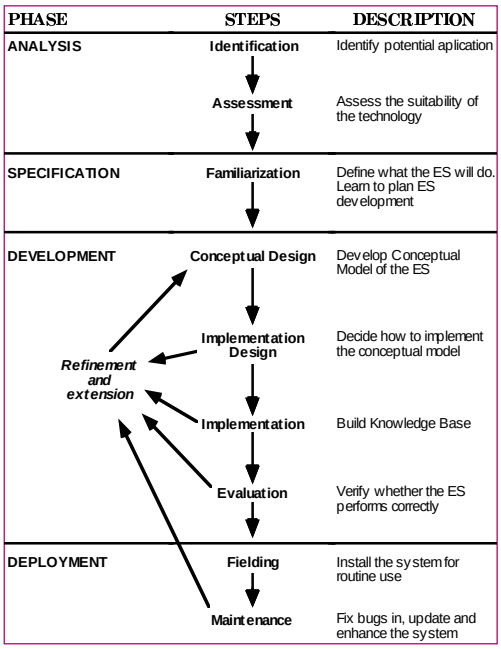

Figure 23 - Expert Systems life-cycle

The Analysis Phase includes the *Identification* of a potential application, taking into account possible motivations for the development of an ES and the characteristics of the task, and also the *Assessment* of the suitability of the ES technology, which comprises the evaluation of technical feasibility, of the balance between benefits and costs, as well as organisational, social, legal and other implications.

The Specification Phase is devoted to clearly define what the ES will do. It comprises *Familiarisation* with the expert's task, to get a general overview of it, with the aim at specifying the overall functionality of the ES, and decomposing it to produce a plan for the incremental development.

The *Conceptual Design* is the first step of the Development Phase, and consists in acquiring expert knowledge for the current scope, focusing on the data needed to solve the task, how the expert infers new information, and the sequence of steps the expert performs to accomplish the task. The result of this step is an organised body of knowledge, the *Conceptual Model*. The *Implementation Design* intends to specify how the conceptual model will be implemented with the selected ES shell, while *Implementation* is intended to create a working ES according to the implementation design. In advanced cycles of development, the implementation stage may require the integration of the ES into its operating environment. The last step of Development is Evaluation. Roughly, in early cycles of development this involves a simple "unit test" of the added functionality, whereas in cycles preceding deployment a formal validation of the overall performance of the ES is done, and the results indicate whether the ES is ready to be fielded.

The Deployment Phase embodies *Fielding* and *Maintenance*. Fielding may include final integration with other systems (e.g., equipment providing input from sensors), installation of specific equipment (e.g., the hardware in which the ES will run) and users training when appropriate. The goal of Maintenance is keeping the ES usable and used, which may involve routine user support, installing new versions of the ES and starting new refinement and extension cycles when needed.

5.2 Knowledge Acquisition

We may verify in Figure 23 that there is no Knowledge Acquisition phase or step in the Life-Cycle of an ES. This is because KA activities play a role in each step in the system's life-cycle, as we may see in Figure 24.

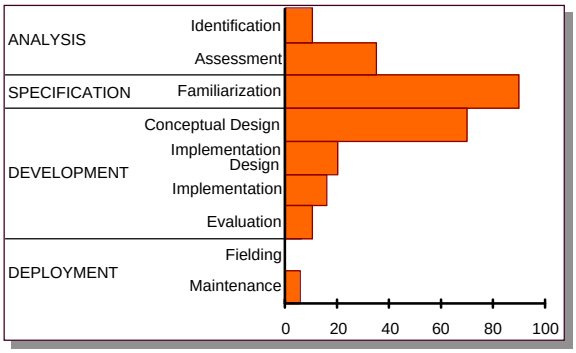

Figure 24 - Typical distribution of percentage of time spent with KA

We may identify two stages in KA activities (Figure 25): *Initial Inquiry*, which runs during Analysis and Specification, and *Detailed Investigation*, during Development and Deployment. In the Initial Inquiry, the goal of the Knowledge Engineer is to get a broad and shallow overview of the task, in order to specify the ES and plan its development. Detailed Investigation is characterised by focus on detail, to get a deep perspective of what the expert knows and how he performs the task. We are going to take a closer look at each of these stages.

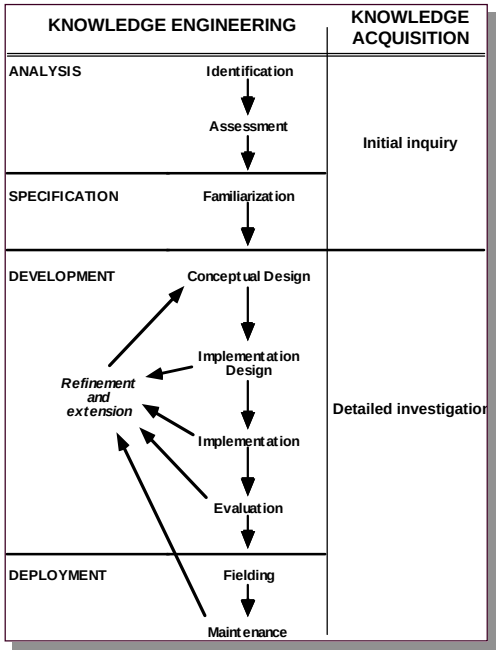

Figure 25 - Knowledge acquisition stages

5.2.1 Initial Inquiry

This stage lays the foundation for the cyclic development. The knowledge engineer works with the expert, the intended users and the project funders to define the specification of the ES and plan the iterative development.

The first question to answer is: what will the ES do? This involves:

i) the definition of the goals which the introduction of the ES is expected to accomplish

ii) the definition of the functionalities it must exhibit

iii) the identification of the users

iv) the specification of the operating environment

v) the identification of the sources of expertise

Once this is clarified, the knowledge engineer can start working with the expert to get a broad understanding of how he performs his task. Depending on the particular situation, this can be accomplished through the direct observation of the expert's activity, meetings where the expert explains the way he performs his task, and/or discussion meetings around specific cases. It is expected that the knowledge engineer ends familiar with all aspects of the task and with the full range of situations in which the task will be performed, without losing focus with unnecessary details: KA will concentrate on deepening knowledge during the second stage.

This broadly acquired knowledge will serve as basis for subsequent discussions with the expert, focused on the elaboration of a development plan. The first thing to do is to define possible partitions for the overall scope of the ES. Partitions may be determined along three dimensions:

- by dividing the task in subtasks (e.g., monitoring, control, diagnosis, elaboration of reports)
- by dividing the input data into a set of classes (e.g., data from ProcessA, data from ProcessB)
- by dividing the output data into a set of classes (e.g., control actions for ProcessA, control actions for ProcessB, diagnoses, reports)

Partitions must then be compared according to

- size (amount of knowledge to acquire)
- difficulty (of the task to perform)
- importance (how significant is the partition to the overall goals of the ES)

The results of this comparison are then used for the identification of the development increments, i.e., the partitions that will actually be considered for plan elaboration and will represent the functionalities to be added in each development cycle.

Once the increments are identified, they are ordered to form the plan, according to the results of the comparison. In principle, a higher priority is given to the partitions with less size, less difficulty and more importance. When appropriate, the funders or their representatives may be asked to participate in this process.

To complete the plan, the development time of each increment must be estimated.

5.2.2 Detailed Investigation

In each cycle of this stage, the KA work is centered around a limited set of functionalities, the *current scope*, according to the development plan. The sequence of activities which are to be performed is:

i) Acquire knowledge for the current scope

ii) Organise knowledge

iii) Create/update conceptual model

iv) Integrate knowledge in KB (not a KA-specific activity)

v) Evaluate system's performance

Actual *knowledge acquisition* comprises working with the expert to precisely understand all the details and facets of his problem-solving activity. To help this work, it is advantageous to consider different *kinds of knowledge* and focus each KA session in one of the kinds. A convenient knowledge classification for this purpose is:

i) Strategic Knowledge, which describes the steps taken by the expert when performing the task

ii) Inferential (or judgmental) Knowledge, which describes how the expert deduces new facts and hypothesis from the available information

iii) Factual Knowledge, which describes the attributes used by the expert when performing the task

Acquiring *strategic knowledge* requires a deep understanding of the task steps, which may be attained through detailed task descriptions and detailed discussions around specific cases, focusing on strategy.

Inferential knowledge requires the identification of each reasoning step performed by the expert. Detailed case discussions focusing on inference are valuable means for this purpose. Particular attention should be devoted to the *use of uncertainty*, and to find *hidden assumptions*, i.e., factors which the expert neglect to mention because they consider them as implicit or obvious.

Acquiring *factual knowledge* comprises the identification of attributes and the definition of their ranges of values. Input data involved in the reasoning steps, as well as final and intermediate conclusions, may constitute the starting points for this work. Sessions devoted to factual knowledge may involve reviewing and discussing specific cases, as well as notes from sessions devoted to inferential knowledge.

The collected knowledge must now be conveniently *organised* to facilitate the design and the implementation of the ES. A thing to do is to organise the records of interviews and meetings to enable a quick access to every peace of information (e.g., creating indexes organised by topic). Also, a *Project Dictionary* of standard terminology used by the development team should be created. This document is important to guarantee the consistency of the terminology and concepts among the team members, and must be kept up-to-date during all the development. Terms from the standard project vocabulary and terms with non-obvious definition should be included in the dictionary. Other document to create and keep up-to-date is a *Case Library* containing the collection of case reports, which are written accounts of the intended behaviour of the ES for each specific relevant case discussed with the expert. Besides being an important help to KA, this library may also be used later during evaluation.

After organising the acquired knowledge, there is the need to do a careful analysis of each piece of knowledge, to clarify its function within the knowledge base. The result of the analysis will be a *Conceptual Model*, which is a set of *intermediate representations* for the three kinds of knowledge: strategic, inferential and factual. Intermediate representations are notations and diagrams which allow the representation of each piece of knowledge according to its function, and are independent of the specific representation formalisms available within the ES shell adopted for implementation. Examples of intermediate representations for strategic knowledge are: flow charts, functional-decomposition trees and decision trees. For inferential knowledge, we may use, for instance: decision tables, rules, decision trees and inference networks. Examples of intermediate representations for factual knowledge are: fact tables, taxonomic trees and graphs[9].

The conceptual model provides an intermediate means for representing knowledge which is close to the human being and to the machine: the intermediate representations are easily understood by the expert and the knowledge engineer, and are easily translated to any Shell-specific

[9] Illustrations of the use of these representations may be seen at [27], Chapter 9.

representation formalism. Therefore, the conceptual model constitutes a key tool in the design and maintenance of the ES as it may support both knowledge acquisition and KB implementation.

In which concerns to KA, it serves as a communication medium with the expert, and helps establishing context for discussions. It also supports implementation as it helps the analysis of the requirements for the ES Shell, and serves as specification for the KB.

After the *implementation* of the KB, an *evaluation* of the system's performance must be done. Possible evaluators are:

i) the project expert

ii) funders' representatives

iii) potential users

iv) experts not associated to the project

Evaluation sessions may be focused on two distinct aspects of the system's performance:

i) the results, i.e., the correctness of the ES problem-solving behaviour;

ii) the interaction, i.e., the way in which the ES interacts with the user.

There are two usual forms of evaluation sessions:

i) the demonstration session, where one or more evaluators review the system while it performs a task;

ii) the off-line session, where one or more evaluators review a written record of an ES problem-solving session.

Although this must not be taken as a definitive rule, demonstration sessions are typically more effective when evaluating the interaction, while off-line sessions are mostly used to evaluate results.

6. EXPERT SYSTEMS FOR MONITORING AND CONTROL

Expert systems, offering the possibility of automating the use of knowledge from the best experts for solving problems, have emerged as very promising tools at mid-seventies, and progressively evolved to a mature state during the eighties, when they have left research labs to tackle industrial and commercial applications. Nowadays, ES are found in a broad diversity of fields, from business to science, engineering, and manufacturing. The arising of low-cost commercial Shells running on personal computers and offering advanced integration and interface capabilities with databases, spreadsheets, data-gathering applications and other conventional programs, favoured the spreading of practical applications of ES. A recent work on the evolution of the field indicated an impressive growth, as researchers developed systems to deal with difficult but commercially rewarding problems [28]. The work was based on a survey [29] which revealed that diagnosis is the most frequent type of application problem for ES, and monitoring and control jointly occupy an important position (Figure 26).

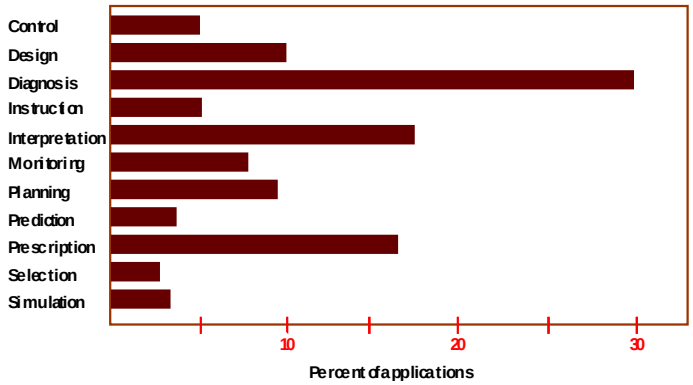

Figure 26 - Expert system applications by problem type[10]

During the last decade, expert systems for control applications were developed and/or fielded in a wide range of domains. One of the first systems referred in the literature was ESCORT [2], whose prototype was tested on a simulation of an oil platform process plant. PICON [13] was the first commercially available development environment for real-time expert systems. G2 [30], derived from PICON and developed by Gensym Corp., is a commercial ES Shell specially tailored to real-time control applications which is currently being applied to several problems. Literature also refers Pilot's Associate [31], intended to give support to the piloting of battle-planes. The RESCU project [32] produced a quality control system. LES [33] was developed to monitor and diagnose the Space Shuttle's launch processing system. IDEA [34] is being currently used in 1,500 Fiat/Lancia/Alfa Romeo repair centers throughout Italy and helps diagnose faults in a variety of automotive electronic subsystems. Sparse [35] is intended to process alarms of power systems. Other examples are MCM [36], DECA [3], IPCS [14], L*STAR [37], PISCES [38], DANTES [39] and AMPERES [40]).

Given the severe requirements for developing expert systems for monitoring, diagnosis and control, particularly when hard real-time constraints are present, as described in Section 2.3, the first fielded applications resorted to high-performance workstations with specially tailored architectures (e.g., PICON). Presently, the technological evolution of personal computers, the growing flexibility, interfacing capabilities and performance of the ES Shells are changing the scenario. Expert systems for process monitoring, diagnosis and control are reaching interesting success levels [41].

Current trends indicate a growing integration of ES with conventional control systems, giving rise to highly autonomous systems, the *expert controllers*, with the ability to incorporate and represent the nonlinearities needed for dynamic systems [42]. Approaches which entirely embed the ES in the control equipment are being explored (e.g., [43]).

One of the known problems of expert systems is that they exhibit a brittle behaviour when they rely solely on heuristic knowledge. The lack of a deeper body of knowledge makes difficult to deal with unforeseen situations. Moreover, as we have already seen, expert knowledge acquisition is a hard task, subjected to errors and misunderstandings; relying exclusively on the expert's knowledge may be acceptable for a great number of situations, but may be too risky for critical applications. Also, explanations generated exclusively from heuristics are often too superficial. To overcome these problems, some of the expert systems presented above embody explicit models of the process under control, and use both forms of knowledge (heuristic and model-based) to solve problems and generate explanations. Explicit models may also facilitate heuristic knowledge acquisition and KB long-term maintenance.

[10] Reproduced from [28].

Other alternative AI techniques are being applied to intelligent monitoring and control. Artificial neural networks, which crudely emulate biological neural networks, exhibit fast response times and learning capabilities. They may be used to improve the control system behaviour by learning on-line [42]. Genetic algorithms are being used to evolve controllers off-line [42].

7. CONCLUSION

Expert systems represent a viable approach to knowledge-intensive problem solving tasks. Their application to monitoring and control may relieve the cognitive overload to which the operators are exposed, which may improve system's performance and safety. A careful assessment of the situation should however been accomplished before starting construction of an expert system, as knowledge acquisition is tricky and time-consuming. Special attention should be devoted to time-response, interface and integration with the control equipment, and availability of expert(s) to allocate to the project.

REFERENCES

[1] J. K. McDowell, J. F. Davis and M. A. Kramer, Knowledge-Based Diagnosis in Process Engineering, IEEE Expert, 6 (3), 65:66 (1991).

[2] P. A. Sachs, A. M. Paterson and M. H. Turner, ESCORT - an Expert System for Complex Operations in Real-Time, Expert Systems, 3 (1), 22:28 (1986).

[3] S. R. Nann, S. Kumara and A. Ray, A Decision Support System for Real-Time Monitoring and Control of Dynamical Processes, International Journal of Intelligent Systems, 6 (7), 739:758 (1991).

[4] E. C. Tacker and M. T. Silvia, Decision Making in Complex Environments Under Conditions of High Cognitive Load: a Personal Expert Systems Approach, Expert Systems with Applications, 2, 121:127 (1991).

[5] M. H. Turner, Real Time Experts, Systems International, 55:57 (1986).

[6] A. Basden, On the Application of Expert Systems, in M. J. Coombs (Ed.), Developments in Expert Systems, Academic Press (1984). [7] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum and J. Lederberg, Applications of Artificial Intelligence for Chemical Inference: the DENDRAL Project, McGraw-Hill, New York (1980).

[8] D. R. Reddy, L. D. Erman, R. D. Fennell and R. B. Neely, The HEARSAY Speech Understanding System: An Example of the Recognition Process, IJCAI-73, 185:193 (1973).

[9] H. E. Pople Jr., CADUCEUS: An Experimental Expert System for Medical Diagnosis, in P. H. Winston and K. A. Prendergast (Eds.), The AI Business, MIT Press, 67:80 (1984).

[10] E. H. Shortlife, Computer-Based Medical Consultations: MYCIN, Elsevier, New York (1976).

[11] P. P. Bonissone and H. E. Johnson, Expert System for Diesel Electric Locomotive Repair, Knowledge-Based System Report, GE Company, New York (1983).

[12] J. McDermott, R1: A Rule-Based Configurer of Computer Systems, Artificial Intelligence, 19, 39:88 (1982).

[13] R. L. Moore, Adding Real-Time Expert System Capabilities to Large Distributed Control Systems, Control Engineering, April (1985).

[14] S. Padalkar, C. Biegl, G. Karsai, N. Miyasaka, K. Okuda and J. Sztipanovits, Real-Time Fault Diagnosis, IEEE Expert, 6 (3), 75:85 (1991).

[15] T. J. Laffey and A. Gupta, Real-Time Knowledge-Based Systems, IJCAI-89 Tutorial Program, Detroit (1989).

[16] B. Wielinga, W. V. de Velde, G. Schreiber and H. Akkermans, Procs. 2nd. Japonese Knowledge Acquisition for Knowledge-Based Systems Workshop, 23:42 (1992)

[17] http://www.laas.research.ec.org/esp-syn/text/5248.html

[18] J. P. Ignizio, Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems, McGraw-Hill International Editions (1991).

[19] P. Lucas and L. Van Der Gaag, Principles of Expert Systems, Addison-Wesley (1991).

[20] F. W. Clocksin and C. S. Mellish, Programming in Prolog, Springer-Verlag (1981).

[21] J. Giarratano and G. Riley, Expert Systems: Principles and Programming, 2nd. Edition, PWS Publishing Company, Boston (1994).

[22] E. H. Shortlife and B. Buchanan, A Model of Inexact Reasoning in Medicine, in Rule-Based Expert Systems, Addison-Wesley, 233:262 (1985).

[23] L. A. Zadeh, Fuzzy Sets, Information and Control, 338:353 (1965).

[24] G. Shafer, A Mathematical Theory of Evidence, Princeton Univ. Press (1976).

[25] I. Graham and P. L. Jones, Expert Systems: Knowledge, Uncertainty and Decision, Chapman and Hall Computing, London (1988).

[26] C. V. Altrock, Fuzzy Logic and Neurofuzzy Applications Explained, Prentice-Hall (1995).

[27] A. Scott, J. Clayton and E. Gibson, A Practical Guide to Knowledge Acquisition, Addison-Wesley (1991).

[28] J. Durkin, Expert Systems: A View of the Field, IEEE Expert, 11 (2), 56:63 (1996).

[29] J.Durkin, Expert Systems: Catalog of Applications, Intelligent Computer Systems, Inc. (1993)

[30] B. Matthews, L. M. Hawkinson, P. Lindenfelzer and R. L. Moore, Process Control with the G2 Real-Time Expert System, ICIEAAIES-88, Procs. 1st Int. Conf. on Industry and Eng. Applications of Artificial Intelligence and Expert Systems, Tullahoma, EUA, 492:497 (1988).

[31] S. B. Banks and C. S. Lizza, Pilot's Associate: a Cooperative, Knowledge-Based System Application, IEEE Expert, 6 (3), 18:29 (1991).

[32] R. Shaw, RESCU: On-Line Real-Time Artificial Intelligence, Computer-Aided Engineering Journal, 29:30 (1987).

[33] E. A. Scarl, C. I. Delaune and J. R. Jamieson, Diagnosis and Sensor Validation through Knowledge of Structure and Function, IEEE Transactions on Systems, Man and Cybernatics, 17 (3), 360:368 (1987).

[34] M. Sanseverino and F. Cascio, Model-Based Diagnosis for Automotive Repair, IEEE Expert, 12 (6), 33:37 (1997).

[35] Z. Vale, A. Moura, M. Fernandes, A. Marques, C. Rosado and C. Ramos, Sparse: An Intelligent Alarm Processor and Operator Assistant, IEEE Expert, 12 (3), 86:93 (1997).

[36] B. D'Ambrosio, M. R. Fehling, S. Forrest, P. Raulefs and B. M. Wilber, Real-Time Process Management for Materials Composition in Chemical Manufacturing, IEEE Expert, 2 (2), 80:93 (1987).

[37] T. J. Laffey, S.M. Kao, J.Y. Read, J.L. Schmidt and S. Weitzenkamp, Intelligent Real-Time Monitoring, Procs. AAAI-88, Vol.1, 72:76, St. Paul, Minnesota, EUA (1988).

[38] E. S. Washington and M. Ali, PISCES: An Expert System for Coal Fired Power Plant Monitoring and Diagnostics, ICIEAAIES-88, Procs. 1st Int. Conf. on Industry and Eng. Applications of Artificial Intelligence and Expert Systems, Tullahoma, EUA, 87:93 (1988).

[39] R. Mathonet, Concepcion et réalisation d'un système expert temps réel. Une étude de cas: un système de détection et de diagnostic de problèmes dans les réseaux d'ordinateurs, Technique et Science Informatiques, 8 (2), 169:179 (1989).

[40] S. C. Lee, L. F. Lollar, C. Paterson and M. Ratliff, AMPERES: a Real-Time Fault Monitoring and Diagnosis Knowledge-Based System for Space Power Systems, Applied Artificial Intelligence, 5, 281:308 (1991).

[41] K. Finke, M. Jarke, R. Soltysiak and P. Szczurko, Testing Expert Systems in Process Control, IEEE Trans. on Knowledge and Data Eng., 8 (3), 403:415 (1996)

[42] K. M.Passino and Ü. Özgüner, Guest Editors' Introduction: Intelligent Control: From Theory to Application, IEEE Expert, 11 (2), 28:30 (1996).

[43] P. Morizet-Mahoudeaux, On-Board and Real-Time Expert Control, IEEE Expert, 11 (4), 71-81 (1996)

BIBLIOGRAPHY

J. P. Ignizio, Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems (McGraw-Hill International Editions, 1991).

A. Scott, J. Clayton and E. Gibson, A Practical Guide to Knowledge Acquisition (Addison-Wesley, 1991).

P. Lucas and L. Van Der Gaag, Principles of Expert Systems (Addison-Wesley, 1991).

Constantin V. Altrock, Fuzzy Logic and Neurofuzzy Applications Explained (Prentice-Hall, 1995).

ADDITIONAL READINGS

A. Gonzalez and D. Dankel, The Engineering of Knowledge-Based Systems (Prentice Hall, 1993).

B. Hayes-Roth, Intelligent control, Artificial Intelligence, 59 (1-2), 213:220 (1993).

D. T. Pham, ed., Expert Systems in Engineering (IFS Publications, Artificial Intelligence in Industry series, 1988).

M.M. Gupta and N.K. Sinha, eds., Intelligent Control: Theory and Practice (IEEE Press, 1995).

P.J. Antsaklis and K.M. Passino, eds., An Introduction to Intelligent and Autonomous Control, (Kluwer Academic Publishers, 1993).

Special Track on Intelligent Control, IEEE Expert, 11 (2-4) (1996).