THE EXPERT SYSTEMS APPROACH: AN ENGINEERING APPLICATION-ORIENTED PERSPECTIVE

Naief Turki Ibn-Homaid

Assistant professor, Department of Civil Engineering, King Saud University.
E-mail: bnhomaid@ksu.edu.sa

ABSTRACT

The state of development in expert systems indicates they are advancing to become a main stream of applied information technology. In this recent area of applied research, an engineering researcher is often faced with the difficult question of how much background material should be understood in order to adequately assess or apply the technology in question to solve a particular problem. This paper presents a concise original analytical review of the expert systems approach with a view to aiding development of expert system applications in engineering. As a result, some useful guidelines are revealed.

Keywords: *expert systems, knowledge-based systems, expert system development, applications in engineering, guidelines.*

الملخص

ينبىء التطور الحاصل في نظم الخبرة بأنهم يحثون الخطى لكي يصبحوا أحد المناحي الرئيسية في تقنيـة المعلومـات التطبيقية. في هذا المجال الحديث من البحث التطبيقي غالباً مايجابه الباحث الهندسي بالسؤال الصعب بخصوص الخلفية المعرفية اللازمة لتقييم أو تطبيق التقنية موضع السؤال لحل مشكلة معينة. تقدم هذه الورقة استعراض و تحليل أصـلي و دقيق لاسلوب نظم الخبرة وذلك للمساعدة في تطوير التطبيقات الهندسية المناسبة. يتجلى من خلال ذلك الاستعراض التحليلي بعض الإرشادات أو الموجهات المفيدة بهذا الشأن.

1. INTRODUCTION

Artificial intelligence (AI) is a computer science discipline concerned with creating intelligent programs, i.e., programs that behave in manners that would be considered intelligent when observed in humans (Waterman 1986). Research aimed at creating such programs has evolved into systems that address different human cognitive and perceptive abilities. One class of such systems is known as knowledge-based or expert systems. These are AI programs designed to solve problems or perform tasks at human experts levels.

Expert systems represent a new approach to solving certain kinds of problems by computers. In this recent area of applied research, an engineering researcher is often faced with the difficult question of how much background material should be understood in order to adequately assess or apply the technology in question to solve a particular problem. The expert systems approach involves both technological and methodological issues, general familiarity with them cannot be taken for granted at the present. Furthermore, as one would expect in the case of a new and emerging technology, the body of literature dealing with expert systems is large and is marked with proliferation and lack of consistent use of the terms. This situation has not been helped by the ambiguity of some AI jargon.

This paper then presents a concise original review and analysis of the expert systems approach with a view to aiding development of expert system applications in engineering. It attempts to sort through the profusion and jargon to get at the underlying concepts to provide the engineering researcher with an understanding of the principles behind expert systems. The information presented is based on insights gained through keen interest in expert systems since they burst into the scene in mid-1980s, a comprehensive critical review and analysis of the literature dealing with those systems and their potential applications, and actual practical experiences gained through a Ph.D. research conducted to develop an expert system in the domain of civil engineering. The important principles and concepts that underlay the general approach are discussed with an engineering application in mind. Since this is not intended to be a complete or extensive introduction to the field of expert systems, the interested reader is referred to the references for a fuller discussion of the approach.

Following this introduction, section two traces expert systems to their origin in the search for computer intelligence. Section three summarizes useful ways of looking at an expert system including its definitions, features, and characteristics. The architecture and structure of a system along with available tools for building it are also discussed. The process and stages of developing a system are reviewed in Section four. Section five reviews weaknesses of today’s systems and potential capabilities of tomorrow’s. Section six briefly discusses expert system applications in engineering, in particular some difficulties in development of certain engineering applications. The paper concludes with a section highlighting key issues that would be of interest to engineering researchers wishing to apply the technology of expert systems in their respective fields of specialization.

2. ORIGIN: A BRIEF HISTORY OF AI AND EXPERT SYSTEMS

Although the dream of duplicating human intelligence in machines can be traced far back in history, the serious search to create such machines is of a much shorter time span. The summer of 1956 is widely acknowledged as the official birth of the research program. The place is Dartmouth where scientists engaged in thinking about how to create intelligent machines conferred. The term ‘artificial intelligence’ was coined in the course of naming the conference (Parsaye and Chignell 1988). Ideas and deliberations about human intelligence

Top

which were presented at that conference along with the availability of the computer in the late 1940s led some scientists to speculate that the dream of intelligent machines is realizable in the form of computer systems. The chief engineering-oriented goal of AI was set to develop programs that can solve problems ordinarily thought of as requiring human intelligence (Duda and Shortliffe 1983).

Following the Dartmouth conference, AI research was dominated by the belief that human intelligence is largely due to the ability to reason. So in the 1960s research efforts were focused on finding general reasoning methods and using them to create general-purpose problem solving programs. Developing those programs proved to be very difficult and ultimately produced little success. This eventually led to the view that the more general-purpose a program is made to be, the more poorly it appeared to perform on any particular problem (Waterman 1986).

In reaction, AI researchers decided to modify their quest for intelligent programs. Instead of creating general-purpose programs, they concentrated on developing general methods to apply in more specialized programs. More specifically, researchers sought general ways to improve on two key aspects of problem-solving: representation, the process of formulating a problem so as to render it easy to solve, and search, the process of navigating through the space of possible solutions to find an acceptable one (Waterman 1986). General methods to formulate problems as search spaces and techniques to efficiently explore the spaces for solutions were developed in the 1970s. They were then used in programs to solve specific problems, mostly games and puzzles.

This group of programs performed better than their predecessors. Some were quite capable of solving problems at levels comparable to human beings. This led to optimistic projections about their performance when applied to real world problems. Some researchers were quick to point out that the problems these programs dealt with are characterized by uniform structures and small search spaces. No sooner trials to adopt those programs in real life applications began than limits of their power started to show. Ultimately, this led to the view that general problem-solving even when augmented with efficiency search techniques is too weak to solve most of real world problems (Hayes-Roth et al. 1983).

Faced with this limitation, a group of AI scientists chose another tack to make programs intelligent. They embarked on creating a special-purpose program to solve a real world problem using specific knowledge obtained from human experts. This attempt proved successful. It resulted in a very specialized program capable of rivaling expert humans at solving the problem. Similar programs were soon to follow. By the late 1970s, several of them were born, leading to the recognition of the primacy of knowledge in intelligent problem-solving. Thus was the inception of the knowledge-based expert systems approach.

Top

3. WHAT IS AN EXPERT SYSTEM?

Success of the latest attempts at creating intelligent problem solvers led to the realization which many now regard as the major conceptual breakthrough in AI research so far. To be intelligent, a program should be provided with a large amount of high-quality specific knowledge about some problem domain (Waterman 1986). Those intelligent programs are known as 'knowledge-based systems' or 'expert systems'. There is no precise or widely endorsed definition, in the classical sense, of either of these phrases. In fact, very often they are used as synonyms. The purpose of this section is to illustrate what these two terms as a unified concept mean. Before that, there is a useful distinction which needs to be clarified.

A knowledge-based system is a computer program that employs knowledge to solve a problem ordinarily requiring human intelligence (Hayes-Roth 1984; Jackson 1999). If the system achieves high-performance in a problem domain that, for a human, requires years of special education and experience, then it is called an expert system (Hayes-Roth et al. 1983; Jackson 1999). More strictly, an expert system models the knowledge of an actual human expert in a particular domain and solves domain problems at expert levels (Mullarkey 1987). An expert system is a knowledge-based system per se. The former is a special case of the latter. Whereas an expert system uses expert knowledge, a knowledge-based system employs human knowledge, though not necessarily expertise (Hayes-Roth 1984; Jackson 1999). In this sense, expert systems are a super subset of knowledge-based systems.

Some of the early systems are truly expert systems in the strict meaning of the term given above. They were intended to prove that AI techniques can be applied to solve problems at human expert levels, thereby establishing the field of expert systems. Those systems were characterized by the significant resources, human and otherwise, spent to develop them. Most of today's so-called expert systems are more appropriately described as knowledge-based systems. They are useful but hardly rival human experts in performance. The two terms can be used interchangeably provided that the important distinction mentioned above is clear.

3.1 Architecture and Structure

3.1.1. Architecture

A schematic of a rule-based expert system is shown in Figure 1. The basic architecture of consists of three main components: the knowledge base, context, and inference engine. The focal component is the knowledge base where knowledge specific to a particular domain of application is stored. The bits and pieces of domain knowledge acquired from a human expert or some other source must be appropriately formulated for use by other parts of the system for problem solving purposes.

Top

The context, also referred to as short-term or working memory, is a transient and dynamic part of the knowledge base. It holds factual knowledge about the specific problem the system is currently trying to solve. Initially, the context contains the given facts that define the case's parameters. As the system reasons, the context expands to include information derived in the course of solving the problem at hand.

Upon solving the problem, the context comprises the given and intermediately inferred facts as well as the solution.

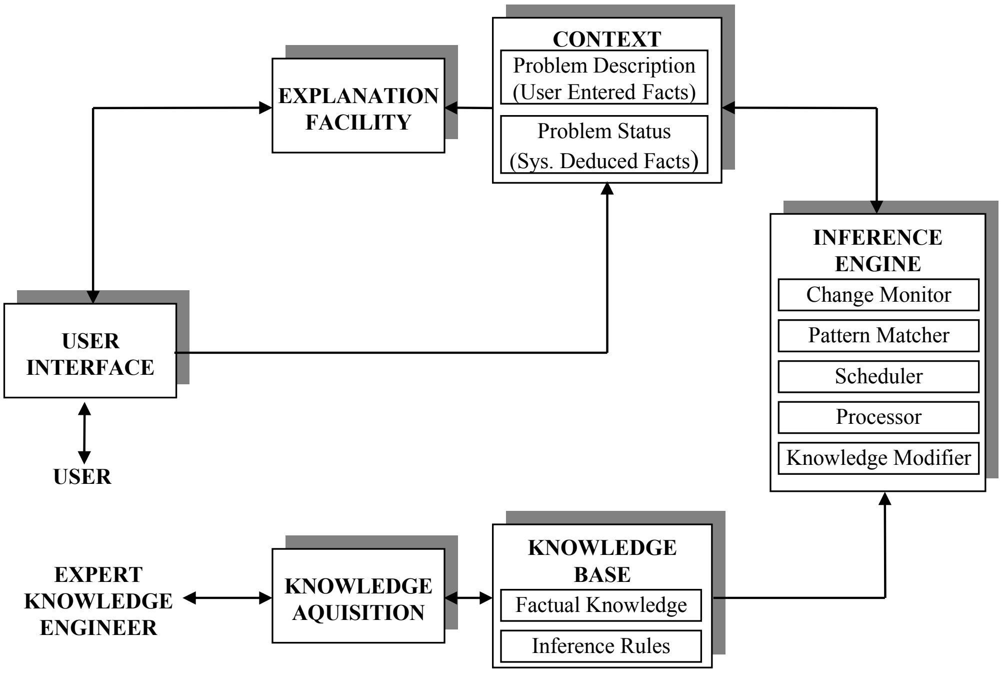

Figure 1 Components of an Expert System.

The component of the system which performs the reasoning task is known as the inference engine. It contains general problem-solving knowledge. The engine typically consists of a change monitor; detects changes in the context requiring attention, a pattern matcher; compares the context and the knowledge base, a scheduler; decides which action to execute next, a processor; fires the required actions, and a knowledge modifier; modifies the knowledge base as specified by the executed actions (McGartland and Hendrickson 1985). Another part not yet as common is called a consistency enforcer which modifies previously drawn conclusions when their bases of support are altered (Hayes-Roth et al. 1983; Jackson 1999). These interrelated parts operate on the knowledge base and the context to solve domain problems.

A complete system will also include three other highly desirable components: the user interface, explanation facility, and knowledge acquisition module. The system communicates with its user through the user interface. In relation to conventional user interfaces, this is a highly interactive and transparent one. Domain-oriented language and means are used to exchange information with the user. Additional unconventional capability required of this interface is to allow the system to provide the user with explanations and justifications of its behavior.

The explanation facility equips the system with the ability to explain its reasoning and justify its actions. On request by the user, a system can ideally answer ‘how’ and ‘why’ questions. The explanations provided by most current systems consist of a backward trace of the steps used to arrive at a particular conclusion. More advanced systems can additionally explain why a certain conclusion was not reached and justify their requests for additional information. The explanation facility is also useful as a debugging tool during system development.

The last and least developed component is the knowledge acquisition module. It is meant to allow initially entering knowledge into the system and later enlarging and maintaining the knowledge base. At present, this component exists rudimentarily in the form of a knowledge base editor. The editor facilitates entering already formalized knowledge. Although, the ultimate goal for this component is to allow knowledge obtained from some source to be added directly, this goal has not been achieved at the present time. Current research on automating knowledge acquisition, carried out in another field of AI, i.e., machine learning, needs to produce more results before the knowledge acquisition module becomes a full fledged component of expert systems.

3.1.2. Structure

To produce an expert system structure, conceptual design decisions must be made at least at two levels. There are no hard and fast rules for making those decisions. At either level, many design options are available. However, the design principles developed to date are not sufficient enough for a structured approach that characterizes the choices in terms of a particular problem domain. As a result, design of an expert system has to proceed pragmatically.

Hayes-Roth (1984) provides a framework that summarizes most of what is known about high-level system design. The framework relates design principles and high-level prescriptions in terms of problem characteristics. Essentially, the most typical structure is presented and various embellishments are added to it as problem complexity increases in various dimensions. Problem characteristics that are considered significant in this respect are the size of the search space and the nature of the available data and knowledge.

Decisions must also be made at a lower design level. The key questions here are how to represent and reason with knowledge. The answers, respectively, determine the structures of the knowledge base and inference engine. A knowledge representation formalism(s) and a

Top

problem-solving strategy(ies) need to be selected. Available techniques for representing knowledge include facts, rules, frames, objects, semantic networks, and formal logic. The most commonly used problem-solving strategies are forward-chaining (data-driven) and backward-chaining (goal-driven). Maher and Allen (1987), Dym and Levitt (1991), and Jackson (1999) provide a review of some common strategies. To complete design, various decisions need to be also made regarding the other architectural components of a system. The most mature and common structure today is the production (rule-based) system's. Where knowledge is represented as rules (productions) and a backward-chaining strategy is, typically, employed. Structural differences have been reflected in the available system-building software tools through specific design choices. Thus, in applied expert systems research the task of conceptual design becomes largely a matter of identifying appropriate techniques and then selecting a tool which implements them.

3.1.3 Expert Systems Building Tools

Expert systems techniques have been implemented, to a lesser or greater extent, in tools for building the systems. Many of those tools are commercially available. A catalogue of most of them can be found in Waterman (1986), Jackson (1999), and Darlington (2000). They vary in many respects. In terms of sophistication, they can be divided into four somewhat overlapping major categories: AI programming languages, general-purpose representation languages, expert system shells, and special-purpose development environments (Mullarkey 1987; Darlington 2000).

The most basic tools are the symbolic languages of AI. The most used are Lisp and Prolog. These languages are distinguished from the traditional ones by the special features offered for representing and manipulating symbolic descriptions. Those features are the low-level primitives in knowledge-based programming. AI languages provide a developer with complete flexibility in designing and implementing a system. The second category refers to higher level, typically Lisp-based, languages for knowledge representation. They can be classified according to the four major paradigms of knowledge representation: rule-based, frame-based, object-oriented, and logic-based. A typical tool in this class offers one or more ready-made structures for representing domain knowledge. As with AI languages, the developer must implement the rest of a system's structure from basic constructs.

Expert system shells are frameworks that aid in rapid development of expert system applications. This group of tools normally consists of skeletal systems, classical expert systems stripped down of their knowledge bases, and the direct commercial derivatives of those systems. They are simply expert systems with empty knowledge bases. A shell usually offers a ready inference engine complete with built-in support facilities (i.e., a user-interface, an explanation facility, and a knowledge acquisition module). A developer can build a system from the shell by filling the empty knowledge base with domain-specific knowledge. A shell reduces and constrains the design options available. Most restrictions arise from the

Top

predefined structures of the inference engine and the knowledge base. Provided a shell is used for a problem similar in type to that for which a shell was originally designed, the shell can speed and simplify the application development process.

Finally, the special-purpose environments denotes tools designed to aid in developing expert systems for different domains and types of problems. Expert system development environments are hybrid (distinguish them from shells) in that they contain different structural mechanisms that implement many expert system techniques. Development environments are also distinguished in that they usually require dedicated AI hardware (e.g., Lisp machines) to run on. A typical environment provides multiple structures for knowledge representation, inference, and control. It also provides extensive development aids including a user interface building facility, an inspector, a browser, editors, and a debugger. As such, an environment offers a great deal of flexibility in designing a system. The structure of a system can be produced by nominating and combining appropriate mechanisms and facilities.

3.1.4 Selecting an Expert System Building Tool

Selection of the right tool to build an expert system is a difficult and critical decision. The decision is difficult in part because many criteria are involved and in part because an appropriate tool needs to be chosen at an early stage, where, especially in the case of a first time developer, the issues involved in representing and applying knowledge of the domain of interest are not well defined. Better understanding comes from experimenting with the tool as well as deeper analysis of the domain knowledge. The choice is critical because a tool cannot only speed up but also facilitate or hinder system development. Two approaches to choosing a suitable tool are common. One is more of a hacker approach whereas the other is more principled.

The first approach employs the so-called consultation or problem-solving paradigm, a conception of how problems of a particular type are solved in different domains. A particular paradigm is a good indicator of what techniques should be used to represent knowledge, perform inference, and control reasoning. A set of potential tools can be identified by matching the consultation paradigm of the problem of interest with those of available tools. The second approach relies on characteristics of the problem domain to identify solution features from among the problem-solving techniques of expert systems. The desired features are then used to identify a set of potential tools (Hayes-Roth et al. 1983; Waterman 1986; Darlington 2000). Neither approach goes as far as recommending a particular tool. Rather, the strategy is to identify a set of potential tools based on necessary tool features and then select a particular tool from the set on the basis of other attributes or considerations.

Many criteria are involved in the choice of an appropriate tool. They are discussed in Gevarter (1987) and Mullarkey (1987) in general and in Gowri et al. (1988) and Moselhi and Nicholas (1988) as they relate to specific engineering domains in particular. The predominant

Top

consideration is the selection of a tool whose built-in structures closely match, or rather allow a close match, with the characteristics of the problem at hand.

3.2 Features and Characteristics

3.2.1 Main Features

An expert system tries to emulate expert problem-solving behavior in a specific problem domain. To do so, the system needs to have the domain knowledge which underlies that behavior. This knowledge is usually of two sorts: public and private (Hayes-Roth et al. 1983). Public knowledge includes definitional, factual, and theoretical information and is widely available in published literature on the subject of application. Expert knowledge contains more than just this public portion. A human expert often relies on private knowledge to solve or simplify a problem in his domain. This latter portion consists largely of heuristic rules and tend to be held privately by an expert or his organization. Building an expert system requires both sorts of knowledge to be uncovered, gathered and formalized.

Knowledge of either sort does not come neatly, pre-assembled, or ready for use. It must be first analyzed and then represented in a form the system can use to solve a problem. Therein lies most of the beneficial features of an expert system. It provides a new alternative for accumulating and codifying knowledge about a problem domain. Relative to traditional means such as textbooks and algorithmic programs, knowledge collected and encoded in an expert system is more explicit, accessible, and useful. Besides the direct benefits obtainable from activating knowledge for problem solving, the system makes domain knowledge, especially private knowledge, readily available for test and evaluation. This in turn fuels the process of refining and improving domain knowledge. "*An expert system acts as a systematising repository over time of the knowledge accumulated by many specialists of diverse experience. Hence it can and does ultimately attain a level of consultant expertise exceeding that of any single one of its 'tutors'*" (Michie 1980 p. 369).

Another feature of an expert system concerns the way it stores and processes knowledge. Domain knowledge in the system is stored explicitly rather than implicitly or abstractly as in algorithmic programming. This provides for a most distinguishing feature; namely transparency. Unlike the 'black box' nature of algorithmic programs, an expert system is able to make its reasoning explicit merely by retracing its problem-solving steps. Knowledge of a domain is encoded in human-like modular form, typically as individual facts and rules each representing an independent chunk, and separated from the means for its use. How that knowledge is used to solve domain problems is the responsibility of the inference engine. Depending on the current problem-solving status in the context, the inference engine selects relevant pieces of knowledge from the knowledge base and regulates the order in which they are applied in accordance with its control strategy. Thus, in contrast to the rigid and static control structure of an algorithmic program, the control mechanism of an expert system is flexible and dynamic. The separation of domain knowledge from control has several beneficial consequences. The system can be developed incrementally over an extended period. Stored knowledge can be modified and new knowledge added without influencing, at least in principle, the control structure.

A final attractive feature of an expert system is what Waterman (1986) calls its predictive modeling power. The system serves as a knowledge-based problem-solving model in the particular domain of application. A user can change a problem's parameters and observe their effects. The user may also experiment with the various structures of the system. This model has the unique advantages of being able to explain how the changes led to the effects and closer to the way human experts in the domain solve problems.

3.2.2 Basic Characteristics

AI researchers have a more sophisticated view of expert systems. To them, an expert system is a computer program defined by several basic characteristics. These properties are briefly reviewed here based on a similar and more detailed discussion elsewhere (Brachman et al. 1983; Waterman 1986; Jackson 1999; Darlington 2000).

An expert system strives to solve a problem as well as human experts. To behave like an expert in this respect means producing good solutions efficiently (Brachman et al. 1983). This requires the expertise a human uses to solve the problem. The emphasis of the work in expert systems so far has been on capturing the heuristic aspect of this expertise. The system employs symbolic reasoning and representation to solve a problem. It uses symbols to represent concepts and states in the domain, uses symbol structures to represent relations among the concepts, and reason by manipulating the symbols.

Another quality the system should have is robustness. It should behave intelligently or degrade gracefully when presented with problems beyond its scope or when given erroneous, inconsistent, or incomplete data or knowledge (Waterman 1986). To attain this quality the system needs to have deep and broad knowledge about its domain. It also needs general problem-solving methods to use this knowledge when its heuristics fail.

An expert system deals with a complex problem. AI researchers believe that a problem has to be complicated enough to be a good candidate for an expert system application. Problem complexity tends to follow from the nature of expert tasks. Computationally, a complex problem is often interpreted as one that does not have tractable or pure algorithmic solutions, or generally, one which is not amenable to such an approach. The system should have a problem reformulation capability. It should be capable of taking a problem stated in lay terms and transforming it into a form amenable to processing by its heuristic rules.

An expert system should possess knowledge about itself, that is, it should know about its structure and operation. The system needs this 'meta-knowledge' (i.e., knowledge about

Top

knowledge) to reason about itself in various forms not the least of which to explain its behavior. A final characteristic concerns the types of problems. Expert systems have been used to solve problems that span the spectrum of problem types (Dym and Levitt 1991) and involve at least one of the following tasks: diagnosis, interpretation, monitoring, prediction, instruction, planning, design, and control (Jackson 1999; Darlington 2000).

It is noted here that an expert system fully exhibiting all of these characteristics does not yet exist. The characteristics represent main defining dimensions AI researchers have used to characterize a true expert system. Existing systems vary with respect to these dimensions in terms of both the extent of coverage as well as degree of achievement. In fact, as the same researchers point out (Brachman et al. 1983), no existing system comes close to achieving the goal on more than one dimension.

With the preceding discussion of features and characteristics in mind, the attributes which differentiate expert systems from other types of programs can be stated. Maher and Allen (1987) and Darlington (2000) list some of the characteristics that distinguish expert systems from conventional programs, whereas Darlington (2000) contrasts expert systems and decision support systems.

A final way of understanding what are expert systems is to compare them to their human counterparts. More specifically, to contrast the natural expertise as it exists in human beings with the artificial one of expert systems. Waterman (1986) and Darlington (2000) list the advantages and disadvantages of artificial expertise relative to those of the natural one.

4. CONSTRUCTING AN EXPERT SYSTEM

The new expert system technology has not advanced to the stage where a well-defined detailed process for developing a system can be articulated. For those wishing to apply the technology, there is pragmatic advice emerging from consensus based on experience to date. The process of developing an expert system is known as knowledge engineering. It has many similarities with the conventional software development process, i.e., software engineering. There are, however, some distinctions.

Two roles are traditionally distinguished in expert system development: that of the domain expert and system developer. The role of the domain expert is to provide the expertise required to solve domain problems. The system developer is historically an AI researcher referred to as knowledge engineer. His role is to assess suitability of the application, acquire knowledge, and build the system.

At the heart of engineering knowledge of a domain is the complex process called knowledge acquisition: extracting, organizing, and structuring domain knowledge for use by the system to help solve domain problems (Waterman 1986). The process involves a collaborative effort between the domain expert and knowledge engineer to uncover the expert knowledge and

Top

express it in a computational form. For a variety of reasons (Stockley 1987), knowledge acquisition is widely acknowledged to be the bottleneck in system development (Jackson 1999). The currently popular case-based reasoning is a promising research direction for widening this bottleneck.

A distinguishing characteristic of expert system development process concerns its iterative evolutionary nature. A system is developed by an incremental prototyping methodology. A first prototype is produced usually for a portion of the problem considered. This is a demonstration prototype normally used in two ways: to assess the possibility of applying the technology to the problem at hand and test the effectiveness of design decisions. A prototyping cycle can then be entered into to increase the system performance and coverage of the problem. In terms of stages, an expert system evolves from being a demonstration prototype to research prototype, field prototype, production prototype, and commercial system (Waterman 1986).

Several phase models of the process of developing an expert system project have been suggested (e.g., Rehak and Fenves 1985; Parsaye and Chignell 1988; Darlington 2000). The most widely mentioned is the one presented by Buchanan et al. (1983) and Jackson (1999), in which five highly interrelated major phases in the process are described: identification, conceptualization, formalization, implementation, and testing. Identification involves identifying the domain expert who will collaborate in constructing the system, defining the problem and its scope, identifying its characteristics and sub-problems, determining the generic reasoning tasks involved, identifying the required resources and knowledge sources, and setting the goals or objectives of constructing the system.

During conceptualization, the knowledge engineer interacts with the human expert to extract domain knowledge and characterize the problem-solving process. Other sources of knowledge such as textbooks, handbooks, manuals, and reports are also utilized in this endeavor. Key domain entities, their attributes, and their relations used to describe the problem and its solution are made explicit. Subtasks, information flow, constraints, and strategies are also identified. The totality of this information represents a conceptual model of the problem domain.

This model is mapped in the formalization phase into a more formal one based on known means for representing and processing knowledge. The knowledge engineer identifies likely needed techniques for representing knowledge, performing inference, and controlling reasoning. The knowledge engineer then selects an appropriate tool for the problem from those available for building expert systems. In implementation, the various ingredients of the problem-solving expertise made explicit during the conceptualization phase are organized and structured in terms of the techniques seemed appropriate. This formalized knowledge is then encoded in the tool chosen to construct the system. The result is an executable prototype program.

Top

Testing involves evaluating the performance of the prototype system. The program is first tested on two or three examples to ensure that it can run from beginning to end. It is then presented with a variety of test examples for further probing. The tests will almost certainly reveal weaknesses and deficiencies in the prototype. The knowledge engineer revises the prototype to overcome the shortcomings. This could entail refining the knowledge base, adjusting the control flow, reformulating the concepts and relationships, redesigning the knowledge and control structures, and even redefining the scope of the initial problem.

Expert system development is not a one-pass or sequential process. Except for the fact that problem identification occurs first and testing last, there is constant jumping between and iteration within the development phases before the initial prototype is produced. The prototype will show if the expert systems approach is appropriate for the problem considered. If it is, a gradual and protracted development cycle can be entered into to broaden and deepen the system coverage of the problem.

5. LIMITATIONS AND EXPECTATIONS OF EXPERT SYSTEMS

Current expert systems are sometimes referred to as first generation. Most of them especially those in routine use today are rule-based, their knowledge is represented uniformly as rules. When assessed with respect to expertness' dimensions that AI researchers feel a system should be expected to exhibit, first generation systems are acknowledged to fall short on a number of the dimensions (e.g., Brachman et al. 1983; Steels 1990). This section reviews some of the main shortcomings and present research attempts at overcoming them. Although expert systems have been used to solve a variety of problems, most success has been achieved more in certain types of problems than others. In terms of the derivation-formation spectrum of problem-solving tasks (Dym and Levitt 1991), expert systems have shown to be more successful and easier to develop for problems falling toward the derivation (e.g., diagnosis and selection) than the formation (e.g., planning and design) end of the spectrum.

First generation systems tend to be narrow; they lack the breadth of knowledge domains experts have, brittle; they degrade rather sharply at the edges of their knowledge and do not recognize their limits, and shallow; they rely on purely heuristic knowledge. Often those heuristics are high-level rules of thumb an expert garnered from many years' experience or compiled from first principles in a domain. The high-level character of expert rules affords a system with reducing its search space thereby producing its solutions efficiently. The efficiency, however, comes at the expense of the system's ability to explain itself and perform robustly (Brachman et al. 1983). As each rule is typically an abstraction of many basic principles and inference steps, a system using high-level rules cannot explain its results in terms of the fundamentals of its domain. Furthermore, the more basic inferences are condensed in one large inferential leap, represented by a high-level rule, the more fragile that leap tends to be. The ability of a system to recognize small variations in its inference patterns diminishes as the level at which those patterns are expressed increases. Small differences that

Top

the system otherwise can recognize, through examination of basic inference steps, and therefore match against go undetected. In effect, a near mismatch is considered as a complete one, contributing to the fragility of the system. Other well known shortcomings of contemporary systems are that they use relatively stylized input-output languages, require qualified users for successful operation, and they lack commonsense. The so-called second generation or deep expert systems are intended to overcome these shortcomings mainly through the use of 'self-knowledge' and 'deep' reasoning (Steels 1985).

Some researchers believe that self-knowledge to be the most potentially important and innovative quality a system can have. Current systems use self-knowledge in providing basic explanations and justifications of their behavior. Usually, this involves some sort of rule-tracing as mentioned earlier. Yet simply displaying the rules invoked in the course of solving a specific problem instance is the least adequate form of explanation. To truly act like a human expert in this regard, a system needs to explain the rationale behind its decisions and even to tailor its explanations to perceived needs of its user. This calls for linking the heuristics with their underlying fundamental principles as justifications and constructing and tailoring of explanations out of those principles. Besides explanation, other potential uses of self-knowledge relate to a system's capability to modify itself by, for instance, restructuring and reorganizing its knowledge base (Lenat et al. 1983).

A deep system has more understanding of its heuristics in the form of a model of deeper principles of a domain. It also has the ability to utilize either kind of knowledge (i.e., surface or deep) as the situation warrants (Steels 1985). As such, it has the potential to degrade gradually at the periphery of its knowledge, know when a given problem is beyond its scope or capability, recognize erroneous data or knowledge, check consistency and completeness of its knowledge base, and even to learn from its experience. A deeper and more self-knowledgeable system promises also to widen the bottleneck of knowledge acquisition and tune its interface to the qualification of a user.

The shortcomings of contemporary systems are the subjects of on-going mostly basic research and they characterize limits of what can be achieved with currently available tools and techniques. Despite its limitations, current technology has resulted in systems that solve complex practical problems. Some of those systems are quite successful, performing in carefully selected narrowly defined problem domains at levels that truly rival those of human experts. The additional techniques currently under investigation in the field of expert systems, i.e., truth maintenance systems, belief revision, case-based reasoning, fuzzy logic, and dependency networks, and other fields of AI, i.e., natural language processing and machine learning, may lead to more versatile and powerful future systems. For the present and from an application point of view, a developer should be content with a less ambitious view of expert systems than what AI researchers have in mind. Duda and Shortliffe's (1983 p. 266) assertion that "*The goal of expert systems research is to provide tools that exploit new ways to encode and use knowledge to solve problems, not to duplicate intelligent human behaviour in all its*

Top

aspects. The challenge at this stage of expert system development is therefore to constrain the problems addressed in realistic ways to allow useful solutions to real-world problems." appears to still hold true.

6. APPLICATIONS OF EXPERT SYSTEMS IN ENGINEERING

Interest in applying expert systems technology in engineering has been growing at a phenomenal pace since its first introduction in the mid-1980s. This is not surprising given that so many of engineering problems are highly dependent on experiential knowledge for successful solutions and these systems promise to make that knowledge widely available. A staggering number of papers and research studies that discuss the relevance and potential of the technology in this area have been reported. As many have reported on the development of expert systems that address the various problems of the area. For instance, Kempf (1989) of the Knowledge Applications Laboratory of Intel™ Corporation gives 160 references on the applications of expert systems in manufacturing planning and scheduling alone, covering only process planning and production scheduling. It is not the intention here to present a state of the art review of the applications of expert systems in engineering. This is in part because, the number of applications or research projects is overwhelming and in part because good reviews are available in the literature. One measure of this can be gained from searching the internet with the subject of expert systems and consulting recent conferences such as the *6th International Conference on the Application of Artificial Intelligence to Civil Engineering* (2002) and the upcoming *7th International Conference on Intelligent Engineering Systems* (2003). Rather the intention is to mention two major sources of difficulty in applying current expert systems technology to certain engineering problems.

Engineering planning, design, and control processes are often characterized by lack of complete and accurate data. Due to incomplete input information at the start, the reasoning process is not continuous. Progressive availability of data necessitates incremental development of solutions. This means that the reasoning process has to be interrupted until required information becomes available. These aspects create a situation whereby the problem-solving status in the dynamically evolving context has to be known over time. In light of the transient character of the context in existing expert system shells, this requirement represents a perplexing hurdle to overcome.

Furthermore, these aspects mean partial solutions, or solutions of differing states, exist most of the time during the reasoning process. Thus, the concern is not only with how to perform but also what has been performed, what needs to be, and when. The answers to these questions are facilitated by the so-called control knowledge. This is a conceptually distinct portion of knowledge of a domain. It refers to how and when the operative domain knowledge is used, a form of meta-knowledge. Capturing control knowledge explicitly is more a characteristic of the second than the first generation of expert systems. First generation systems encode this kind of knowledge implicitly in the form of the reasoning and control strategies of the inference engine. As the size or complexity of the search space increases, however, more of the control knowledge in a problem domain needs to be incorporated explicitly to help focus the search for solutions. It is as if the states of the system are monitored to bring the relevant pieces of knowledge to bear at the proper context.

The other major complicating characteristic is changes, often caused by data uncertainty. Their effect is that evolving solutions need to be revised based on the most current information. In expert systems terms, this means that the reasoning involved is 'non-monotonic'. Available techniques, collectively referred to as truth maintenance or belief revision system or facility, for dealing with non-monotonicity of reasoning are less developed. A sign of this is there seems to be no expert system shell expressly designed for planning or design applications.

The discontinuity and non-monotonicity of reasoning are complicating characteristics that tend to limit the potential of expert systems to engineering diagnosis and selection problems. It might be added that future continuation of this research direction, in light of expected advances in expert systems technology, for the other engineering problems may be worthwhile.

7. SUMMARY AND CONCLUSIONS

Expert systems represent a new approach for dealing with important and difficult engineering problems. That is, practical problems which cannot be solved algorithmically because they, by their nature, resist the algorithmic approach or are not understood well enough to reduce to mathematical models or equations. In such problems, conventional programs offer little help and expert humans are relied upon for satisfactory solutions. Expert systems extend the range of computer applications to such expert problems.

The approach provides practical means for organizing and structuring knowledge of a domain, including the isolated bits and pieces of scattered experiences and practices of the domain, for more practical use and towards better and deeper understanding. Beside their ability to represent knowledge more naturally, the strength of expert systems lies in their selective application of knowledge which derives from their flexible and dynamic control structure. This contributes to the practicality of the approach and renders it perhaps the best available means especially in light of the progressive availability of data and the frequent changes involved in engineering planning, design, and control applications.

Many engineering problems do not belong in the typical class of problems, i.e., diagnosis and selection, where expert systems have scored some success in emulating human experts problem-solving behavior. Rather, they tend to the more complex planning, design, and control problems where expert systems are relatively less successful and more difficult to develop. For these latter problems, knowledge-based as opposed to expert systems are seen as a necessary first step. They are developed toward increased understanding of a problem and

Top

improving its solution in relation to emulating behavior of an actual human expert. It seems to this less ambitious aim that the expert systems approach is being explored in engineering.

Engineering problems characterized by incomplete and inaccurate process data present a challenge for the current expert systems technology. One key aspect of this concerns the need to reconsider developed solutions in light of subsequent revisions. Another is that of maintaining consistency of evolving solutions. Evidently, expert systems techniques for dealing with these reasoning complexity aspects are relatively less developed. Maturity of those techniques will increase the utility of expert systems in engineering.

REFERENCES

1. Brachman, R.J., Amarel, S., Engelman, R.S., et al. 1983. What Are Expert Systems? Building Expert Systems, Hays-Roth, F., Waterman, D.A., and Lenat, D.B., eds. Addision-Welsey, Reading, MA, 31-57.
2. Buchanan, B.G., Barstow, D., Bechtel, R., et al. 1983. Constructing an Expert System. Building Expert Systems, Hays-Roth, F., Waterman, D.A., and Lenat, D.B., eds. Addision-Welsey, Reading, MA, 127-167.
3. Darlington, K. 2000. The Essence of Expert Systems, Prentice Hall, Don Mills, Ontario, Canada.
4. Duda, R.O., and Shortliffe, E.H. 1983. Expert Systems Research. Science, 220(4594), 261-268.
5. Dym, C.L., and Levitt, R.E. 1991. Knowledge-Based Systems in Engineering, McGraw-Hill, New York, NY.
6. Gevarter, W.B. 1987. The Nature and Evaluation of Commercial Expert System Building Tools. *Computer, IEEE*, may, 24-41.
7. Gowri, K., Bedard, C., and Fazio, P. 1988. Evaluation of Knowledge-Based System Development Tools for Building Engineering Design. Proceedings of the Third International Conference on Computing in Civil Engineering, Aug. 10-12, Vancouver, B.C., 430-438.
8. Hays-Roth, F. 1984. The Knowledge-based Expert System: A Tutorial. *Computer, IEEE*, 17(9), Sep., 11-28.
9. Hays-Roth, F., Waterman, D.A., and Lenat, D.B. 1983. An Overview of Expert Systems. Building Expert Systems, Hays-Roth, F., Waterman, D.A., and Lenat, D.B., eds. Addision-Welsey, Reading, MA, 3-29.
10. Jackson, P. 1999. Introduction to Expert Systems, Addison-Wesley, Don Mills, Ontario, Canada.
11. Kempf, K. 1989. Manufacturing Planning and Scheduling: Where We Are and Where We Need To Be. Extended Abstract of an Invited Presentation, *IEEE*, 14-19.
12. Lenat, D., Davis, R., Doyle, J., et al. 1983. Reasoning about Reasoning. Building Expert Systems, Hays-Roth, F., Waterman, D.A., and Lenat, D.B., eds. Addision-Welsey, Reading, MA, 219-239.

Top

13. Maher, M.L., and Allen, R. 1987. Expert Systems Components. Expert Systems for Civil Engineers: Technology and Application, Maher M.L., ed. ASCE, 3-14.
14. McGartland, M.J., and Hendrickson, C.T. 1985. Expert Systems for Construction Project Monitoring, Journal of Construction Engineering and Management, ASCE, 111(3), 293-307.
15. Michie, D. 1980. Expert systems. *The Computer Journal*, 23(4).
16. Moselhi, O., and Nicholas, M.J. 1988. Expert System Tools for Construction Planning and Control. *Microcomputers in Civil Engineering*, 3, 75-80.
17. Mullarkey, P.W. 1987. Languages and Tools for Building Expert Systems. Expert Systems for Civil Engineers: Technology and Application, Maher M.L., ed. ASCE, 15-34.
18. Parsaye, K., and Chignell M. 1988. Expert Systems for Experts. John Wiley & Sons, Inc., New York, NY.
19. Rehak, D.R., and Fenves, S.J. 1985. Expert Systems in Civil Engineering, Construction Management, and Construction Robotics. *Annual Research Review*, Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, 51-66.
20. Steels, L. 1985. Second Generation Expert Systems. *Journal of Future Generation Computer Systems*, 1(4) 213-221.
21. Steels, L. 1990. Components of Expertise. *AI Magazine*, Summer, 28-49.
22. Stockley, J.E. 1987. Knowledge Acquisition for Expert Systems. *Building Cost Modelling and Computers*, 477-489.
23. Waterman, D.A. 1986. A Guide to Expert Systems, Addison-Wesley, Don Mills, Ontario, Canada.

Top