Guest Editors’ Introduction

Expert Systems for Engineering Applications

Duvvuru Sriram, Massachusetts Institute of Technology
Michael D. Rychener, Carnegie-Mellon University

Developments in computer science and engineering methodologies in the 1960's and 1970's provided application program developers with a variety of hardware and software tools. These tools increased the use of computers by engineers, but use has been limited almost exclusively to algorithmic solutions such as finite element methods and circuit simulators.

Many engineering problems are not amenable to purely algorithmic solutions. As Koenig put it, "The engineering method is the use of heuristics to cause the best change in a poorly understood situation within the available resources." To deal with these ill-structured problems, an engineer relies on his own judgment and experience.

Knowledge-based expert systems provide a programming methodology for solving ill-structured engineering problems. Since these systems also provide a flexible software development methodology—by separating the knowledge base from the inference mechanism—they are of increasing interest to the engineering community.

Engineering problems

The kind of problems that engineers face in their day-to-day work can be bounded by the derivation-formation spectrum. In derivation problems, the problem conditions are specified as parts of a solution description. A finite set of possible outcomes exists in the knowledge base, and the knowledge base is used to complete the solution. Essentially, solving these problems means identifying the solution path. In formation problems, conditions are given as properties the solution must satisfy as a whole. Most real-life problems fall between these two categories.

Derivation. Engineers normally encounter the following problems at the derivation end of the spectrum:

Interpretation. The given data are analyzed to determine their meaning. The data are often unreliable, erroneous, or extraneous.

Diagnosis. Identification of a problem area or a fault is based on potentially noisy data. The diagnostician must be able to relate the symptoms to the appropriate fault.

Repair. The faults in the systems are identified, and remedial actions are suggested. Fault diagnosis is the first step in this process.

Monitoring. Signals are continuously interpreted, and alarms are set whenever required.

Simulation. A model of the system is created, and the outputs for a set of inputs are observed.

Control. The data (from sensors) are interpreted, and any deviations from the normal are corrected.

Formation. Typical problems encountered at the formation end of the spectrum are

Planning. A program of actions is set up to achieve certain goals. The actions should not require excessive resources or violate legitimate constraints.

Design. Systems or objects that satisfy particular requirements are configured. This involves satisfying constraints from a variety of sources.

Approach comparison

There are several important differences between the traditional software-engineering development cycle and the cycle for knowledge-based expert systems (Waterman provides an excellent treatment of the latter).

Most software engineering projects assume that the problem is one of implementation rather than design. The rigid specifications and modularization imposed are no longer helpful for projects using knowledge-based engineering systems. There, the project should be thought of as a design problem rather than an implementation issue.

Functional specifications cannot be accurately detailed with expert systems. They change as a wider body of test cases and field problems are covered by the system's behavior. For example, in the development of a system for automating design, the client may not be able to fully specify his needs, which are more like aspirations than specifications.

Domain experts are seldom skilled in knowledge engineering techniques. Hence, continuing interaction with a knowledge engineer is needed until the system has been developed to a nearly final state—that is, until the fine-tuning stage is reached.

The style of program implementation and development is different for expert systems, which are grown incrementally rather than programmed. The program is interactive by nature, and new knowledge units are formulated as the expert uses the program and applies it to new test cases. This contrasts with simply implementing code to meet a functional specification.
prepared in advance of the implementation.

Knowledge engineering tools are rarely exactly suited to a particular engineering problem. More often, they require adaptation and evolve during the knowledge acquisition and implementation process. Engineering problems are inherently diverse and will continue to pose challenges to the builders of such tools.

Higher level program strategies, such as interfaces and interactions among modules, are often represented in the same formalism as the knowledge base. Thus, these aspects of the program also grow gradually and interactively as the program evolves to handle more and more test cases. Exploratory programming environments seem appropriate for these tasks.

Four articles

This issue reports on software engineering aspects of knowledge-based expert systems that involve diagnosis, control, simulation, and design.

Diagnosis. In the first article, “A Qualitative Modeling Shell for Process Diagnosis,” Timothy F. Thompson and William J. Clancey describe the use of an existing shell to develop an expert system for another domain. A system for diagnosing malfunctions in industrial sandcasting, is constructed in the shell environment of Heracles, a heuristic classification shell generalized from the Neomycin medical diagnostic system.

The heuristic classification technique helps develop knowledge-based expert systems for the derivation end of the engineering problem spectrum. The authors show that a well-defined diagnostic procedure and a relational language to state a qualitative domain model are general tools that can be applied to engineering as well as medicine. The authors also used manuals and accessible literature, rather than a human expert, to build the system—a useful knowledge acquisition strategy.

Control. In "An Expert System for Real-Time Control," M. Lattimer Wright and his coauthors touch on a number of practical software engineering issues. Their experimental system for dealing with control problems in military and advanced industrial applications has some unique problems. A hybrid system, Hexscon combines algorithmic real-time control techniques with knowledge-based approaches to higher level reasoning to organize the inference engine at different levels of detail.

Among the Hexscon's design goals are a capacity of 5000 rules in a 512K-byte microcomputer, a response time of 10 to 100 milliseconds, control of about 1000 objects, and continued functioning in an uncertain environment. A prototype has demonstrated these capabilities in several problem domains.

Simulation. In the third article, Yv. Ramana Reddy and his coauthors describe KBS, a knowledge-based simulation system developed as part of a comprehensive, intelligent factory management system. Designed to help managers analyze complex systems, KBS takes advantage of several artificial-intelligence programming paradigms to provide a simulation-based decision support tool.

KBS incorporates a number of features not found in general-purpose simulation environments. One essential difference lies in the use of heuristics to evaluate the output from the simulation program. The frame-based Schema Representation Language, along with graphics utilities, provided the environment needed for system development.

KBS offers facilities for automatic generation of model scenarios, goal-directed instrumentation, rule-based diagnosis, cause-and-effect analysis, causal path analysis, and interactive model building. It has been used successfully in simulation applications that include printed circuit board manufacturing, light bulb manufacturing, flexible assembly, and corporate distribution and inventory.

Design. In VLSI circuit design, implications and consequences of a particular design may not become clear until it has been partially—or almost fully—worked out. At that point, the layout may have to be completely reworked.

The complexity of IC design has made it difficult to apply traditional programming methods to the task. Recently, however, artificial intelligence techniques have been used successfully in computer-aided IC design systems. In the final article, Jin Kim and John McDermott describe and evaluate one such system, Talib.

Talib assists the VLSI designer with cell layouts for NMOS technology. Talib's rule base, implemented in the OPS5 general-purpose language, is rich in domain knowledge. Talib also combines planning with
the design process and, most importantly, demonstrates the impact of knowledge engineering techniques on electronic system design.

Following the four articles is a special section on research in knowledge-based engineering systems. It reports work in progress at 13 universities and research centers. Included are projects in fault diagnosis, expert interfaces, structural design, mechanical engineering, electrical engineering, geographic databases, error recovery in robot control and more.

Acknowledgments
We would like to thank S.J. Fenves of Carnegie-Mellon University's Department of Civil Engineering and Raj Reddy of the university's Robotics Institute for providing the environment to compile this special issue. Our thanks also go to Bruce Shriver for giving us the opportunity and the necessary encouragement and to the referees for doing a great job in reviewing the articles.

Duvvuru Sriam is an assistant professor of civil engineering at the Massachusetts Institute of Technology. He has coauthored a number of papers and two forthcoming books on the application of knowledge-based expert systems to engineering problems. He has also organized conference sessions, conducted tutorials in the US, Canada, and India, and is serving as the technical program chairman for the First International Conference on AI in Engineering, to be held in April 1986 in the United Kingdom.

Sriram has a BTech from the India Institute of Technology in Madras, India, and an MS and a PhD from Carnegie-Mellon University. His address is Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139.

Michael D. Rychener is a research scientist at Carnegie-Mellon University with a joint appointment in the Robotics Institute and in the Design Research Center. His research focus is systems that integrate a variety of approaches to problem-solving using the blackboard model. Another interest is artificial intelligence systems that adapt to changing environments and learn from instruction.

He received his BA degree from Oberlin College, MS in computer science from Stanford University, and PhD in computer science from Carnegie-Mellon University.

Rychener's address is Design Research Center & Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA 15213.

References