RAND | RESEARCH BRIEF
NDRI National Defense Research Institute

Information Processing Systems Program

EVALUATING TOOLS USED IN BUILDING EXPERT SYSTEMS

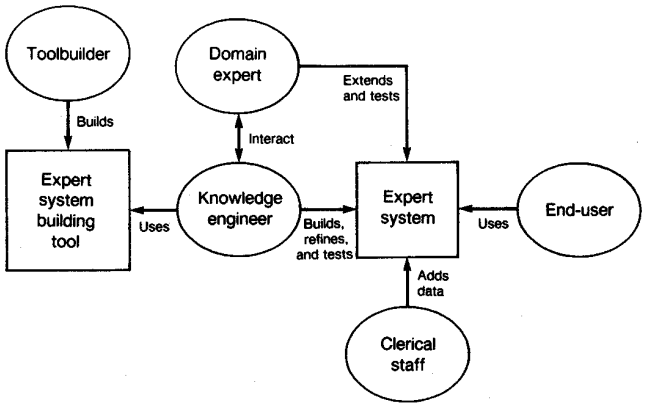
Developments in artificial intelligence have permitted the construction of computerized systems that can apply expert knowledge to help solve real-world problems. Expert systems are now being used in manufacturing resource allocation, scheduling and configuration, military acquisition, and diagnosis and classification tasks.

An expert system is a program that consists of two basic parts—a *knowledge base* and an *inference engine.* The former contains knowledge (facts, rules, strategies, etc.) drawn from experts in the domain of concern (logistics, tactics, etc.). The inference engine is a subsystem that can derive conclusions from the knowledge base, allowing the system to solve problems that were not explicitly programmed. Because they make explicit the knowledge on which their decisions are based, expert systems are usually easier to understand and validate than traditional software.

An expert system is built by a *knowledge engineer,* who interacts with domain experts to encode knowledge, elaborate a prototype system, and iteratively test and refine it (see the figure). Each of these steps is facilitated by the use of special programs or languages, which are referred to as expert system *tools* or *shells.*

Tools for use in developing expert systems are typically large, complex systems in themselves, requiring major investments of time, money, and effort to realize their full advantage. It is thus important that the tools chosen be a good fit for the job at hand. However, choosing appropriate tools is often difficult. A knowledge engineer faces a plethora of tools with different objectives. Obviously, guidelines for evaluating and selecting expert system tools would be helpful.

Under the sponsorship of the Defense Advanced Research Projects Agency (DARPA), information scientists from NDRI reviewed available tools, surveyed tool and system developers, and drew up an evaluation framework. They held workshops for tool and system builders to discuss the framework and learn more about the concerns of those groups. The results of the study are as follows.[1]



Persons involved with expert system development and use

A NEW FRAMEWORK AND METHODOLOGY FOR TOOL EVALUATION

Prior attempts at evaluating tools have been limited in scope and thoroughness. The new framework will help evaluators set out their criteria more explicitly and account for all factors that are potentially relevant to tool selection. It can be applied to tools that have not yet appeared on the market, and it should aid in developing new tools. **The framework calls for identifying and considering each of the following:**

1. **Application characteristics,** i.e., the nature of the problem to be addressed and the project that is to address it. These characteristics include the nature of the problem domain, the available sources of expertise,

[1]R-3542-DARPA, *Evaluating Expert System Tools: A Framework and Methodology,* Jeff Rothenberg et al., The RAND Corporation, July 1987; N-2306-DARPA, *Evaluating Expert System Tools: A Framework and Methodology—Workshops,* Jeff Rothenberg et al., The RAND Corporation, July 1987. Both documents are available through the RAND Publications Department.

the set of development tasks to be undertaken with the tool, and the project's budget.

2. The **contexts** in which the tool is to be used. These are roughly equivalent to the phases of system building, i.e., conceptualizing, prototyping, development, delivery, maintenance.

3. The **tool capabilities**—e.g., handling uncertainty, controlling inference, explanation—that are relevant to the specified application characteristics and contexts. Capabilities are likely to be a more helpful basis for evaluation than specific features listed by the tool developer, because seemingly equivalent features in different tools may provide different capabilities.

4. **Metrics,** or the specific criteria to be used in evaluating the tools. Metrics that may be applicable include cost, flexibility, clarity, efficiency, vendor support, and extensibility (which includes breadth of applicability and ease of integration). The importance of each of the metrics varies with the phase of development.

5. **Assessment techniques,** or ways to apply the metrics to the tools (e.g., to decide what capabilities are actually present). Information sources that are helpful in applying metrics include published lists of capabilities, published test applications (benchmarks), and interviews with other tool users.

The first step in applying the framework is to define each of the items listed above for the case at hand. Next, the evaluator identifies those tools that fulfill the required capabilities and that meet any other absolute constraints, e.g., cost. **The metrics can then be applied via the assessment techniques to evaluate the capabilities of the tools to function in the required contexts, given the application characteristics under consideration.**

The NDRI framework is intended to help in organizing the evaluator's thoughts. The specifics of its application must be elaborated in each instance. At present, the ability of an evaluator to apply such a framework is limited by the availability of relevant information sources. Building and maintaining databases of published case studies or benchmark solutions would require considerable work and cooperation on the part of tool users and vendors. An alternative would be to create an organization to assume this responsibility, preferably one funded by a government agency such as DARPA or by a consortium of tool-user groups.

THE STATE OF THE ART

The research team's discussions with tool builders and users revealed **the pivotal position of software engineering in expert system development.** Building an expert system is as much an attempt to solve questions of representation, integration, debugging support, and so on, as an effort to resolve knowledge- and domain-related issues. Indeed, the failure of most tools to support integration within hardware and software environments is a critical impediment to the construction of expert systems that have to be embedded in other systems, as is the case in some DoD projects.

The overwhelming majority of tool users are convinced that **the tools are well worth the expense and that vendors are generally helpful and supportive. Of course, the tools do have some drawbacks.** For example, many tool users feel that, for most rule-based expert system tools, the indeterminate order in which the rules are invoked makes it difficult to specify intended sequences of events. Also, some tools may be released prematurely; users strongly prefer learning a tool only after it has been freed of bugs. Finally, expert system tools may not be worth the investment where performance speed or flexibility is paramount. Projects that are amenable to algorithmic solutions may always be executed more efficiently in traditional procedural languages, and some projects have special requirements that argue for in-house tool development.

The lessons learned from expert system development are contributing in important ways to software engineering in general. Expert system development represents a shift away from the traditional approach to software engineering that begins with requirements analysis and proceeds through specification, design, implementation, and testing, with little or no mechanism to back up and rethink previous phases. The use of rapid, iterative prototyping with continuous feedback that has been pioneered by expert systems may eventually increase software productivity and effectiveness in general. Certainly, DoD and industry software acquisition standards should be reexamined to accommodate this approach.