A Survey of Expert System Projects

Earl D. Sacerdoti

This paper was originally presented at *Software Development '89,*
San Francisco, California (1989)

A pioneer in commercializing expert system technology, Teknowledge released two so-called "expert system shells" in mid-1984. It soon became apparent that product customers were using these tools in ways that differed from what the developers envisioned. Even internal to Teknowledge, there was considerably controversy over the value of these tools. The debate centered on the tradeoffs between the leverage they provided for certain portions of the system development task and the restrictions they imposed on the ways knowledge could be represented.

By early 1987, over 1,800 copies of this expert system development software had been installed. With such a large installed base, many of the questions being discussed on a theoretical basis could be answered with an objective survey.

With this background, we set out to examine in depth a meaningful fraction of the projects undertaken with our software. We aimed to accomplish a number of objectives. We wanted to understand the users' needs better, in order to focus product enhancement efforts on features that were most needed. We wanted to learn how the products were actually being used. We wished to determine the economic benefits that customers had targeted or were achieving. And we wanted to develop success stories, to help advance both our own business and the general acceptance of this technology. In summary, we set out to do this work in order to influence Teknowledge's product and business strategy for the future.

While Teknowledge subsequently withdrew from the "boxed product" business, the results of this survey are of value both to current and potential users of other "shells," as well as to managers considering the potential impact of expert systems on their organizations.

Overview of the Survey

Over 100 customer sites were contacted to determine the nature of their work with expert systems. The sampling was not random. Larger customers and earlier customers were favored, on the basis that they would have had more time and resources to have devoted to fielding expert

systems applications. The educational institutions using the products for instruction and research were underrepresented, on the basis that they had little rationale to pursue an application all the way to fielding it.

We examined 153 customer projects. Over 25% of these had resulted in a fielded application. Many others were in field test but not yet in regular use by people other than the developers.

Teknowledge's S.1 and M.1.

Teknowledge offered two programming languages for developing expert systems. S.1 was a highly-structured, strongly typed language intended for programming teams to develop larger-scale applications. M.1 had a simple, English-like syntax and was intended for development of smaller-scale systems on a single machine. S.1 was modeled after Pascal; M.1 was modeled after BASIC. S.1 ran on a wide variety of UNIX-based systems. M.1 ran on IBM PCs and compatibles under MS-DOS. Both of these "shells" are fundamentally rule-oriented, with some support for object-oriented programming. The survey sample used M.1 in about a 5-to-1 ratio over S.1, similar to the overall installed base.

Originally. S.1 was written in LISP, and M.1 was written in PROLOG. Both were rewritten in C during 1985, primarily to facilitate integration and to improve performance.

Methodology

Developers were contacted by telephone and we scheduled an interview session. The interviews themselves were generally conducted by phone as well. They generally ran between one and two hours. An interview form was filled in, covering the following points:

- Nature of the application
  - The target organization
  - The target user
  - The role of the system
  - Knowledge required by the system
- System architecture
  - System components
  - Integration and performance issues
- System development and deployment
  - Profile of development effort over time
  - Profile of the programmers
  - Profile of the experts
  - Testing procedures
  - Fielding procedures
- Business case analysis
  - Evaluation criteria
  - Alternative approaches

Costs
Measures of benefits
Organizational obstacles, if any

The data thus gathered was aggregated in several forms for use in product planning, development of a series of seminars on applications of expert systems to various fields, and creation of sales collateral material.

Types of Applications

We identified five basic types of problem-solving that were being applied in a range of functional areas. The diversity of types of applications greatly exceeded our expectations. The chart below indicates the dispersion of both application projects and fielded systems across a broad range of functional areas and a diverse set of problem-solving activities. There was a plurality of manufacturing applications and a preference for a diagnostic problem-solving approach.

	Manufacturing/ Engineering	Service/ Hot Line	MIS	Financial	Government	Other
Diagnosis	**16**	**19**	**2**			
Planning	**14**	**3**	**3**	**5**	**3**	**4**
Design	**18**		**3**			
Interpretation	**6**		**1**	**6**	**5**	**3**
Instruction	**2**				**3**	**1**
Other			**4**		**1**	

Survey Results

While there was great diversity in the experiences of the developers of fielded applications, some general trends emerged:

1. Development and Delivery Hardware

The favored delivery vehicle was an IBM-PC-class personal computer. The second most popular among our samples was the Digital Equipment VAX line. Development projects that began on LISP machines always migrated to conventional hardware prior to fielding, with one exception. (That system was ported to a micro-VAX within six months of deployment.) Projects that began

on conventional hardware were generally deployed on the same hardware that they were developed on.

2. Performance

Expert system performance was generally considered satisfactory. There were even cases where expert systems performed dramatically faster than the less powerful systems they replaced. Times for an individual problem-solving session ranged from about 30 seconds to as much as an hour. The typical problem-solving session lasted only a small number of minutes.

3. Integration

The clearest conclusion derived from the survey is the paramount importance of integrating expert systems with existing computer systems and existing applications. About half of the fielded systems were integrated with file- or database-management systems, graphics interface packages, telecommunications software, or existing applications packages. A majority of the systems in advanced stages of development were integrated as well.

Based on our early experience in the AI laboratories, we expected the focus of the users' efforts to be on knowledge representation, control of the inference process, and other knowledge engineering issues. However, contrary to our original expectations, our users spent at least as much time on traditional software engineering as they did on knowledge engineering. Put another way, fully half the value added in the development of an expert system application is in its integration with the existing environment.

While most expert systems developed in the AI labs followed the model of a question-and-answer consultation, we found that few of the fielded applications did so. They generally used a file to hold the data required for analysis. The file was typically populated through a form-filling front end or via automated access to other online data. Some applications, in fact, weren't interactive systems at all. They received their data automatically from other application software and returned an analysis in the form of an on-line report or file entries.

Whereas expert systems have often been associated with specialized, highly interactive graphics interfaces, we found that most developers preferred to replace our standard interfaces with ones that conformed to existing corporate standards or that were similar to interfaces the end users

were already familiar with. The ability of our software to be embedded within existing user interfaces was critical to many application efforts.

4. The Developers

The typical successful application was developed by a single individual or a small team. Few developers had formal training in artificial intelligence. The typical successful developer had done some reading or learning about the subject on his or her own. However, he or she had not written any AI-oriented code prior to the project.

Many of the developers responsible for successfully fielded expert systems had a history of successfully fielding traditional systems.

5. The Development Effort

There was a very wide range of levels of effort associated with the systems being developed. There was no correlation between level of effort and fielded status. Successfully fielded systems required from six work weeks to six or more work years. Few systems took less than six calendar months from conception to fielding; we have also seen few fielded systems that did not have at least an initial version in the field after two calendar years.

There was also great variance in the amount of effort devoted to application projects by the experts. In some cases the expert's participation was almost perfunctory, with most of the knowledge being gleaned from written materials. Several successful projects had substantially more time devoted by the experts than by the programmers. In these cases, the development of the expert system was used as a framework to guide the development of the expertise. The holes in the knowledge system were due to holes in the corporation's knowledge, and so the expert system project created an agenda for research.

In a significant number of fielded systems, the expert had some programming experience and served as a functioning programmer on the team.

6. Maintenance

A surprising result of the survey was the relatively low effort required for maintenance of fielded systems. The typical system was supported by a fraction of the time of one of the original developers. I believe this is due to the relative stability of the knowledge encoded into most of these systems.

7. Payoffs and Spinoffs

The economic value of fielded applications was often difficult to measure directly. Their benefits included avoidance of future costs, improved quality or timeliness, or lowered training costs for new personnel. Some systems did have directly measurable economic benefits, and these were often substantial----as much as a tenfold payback of development costs within months of fielding.

Other less quantifiable benefits were cited frequently. Many systems serve a secondary function as a training aid. Some developers put extra care into providing careful explanations of the systems' reasoning. This permits users to become less reliant on the expert system as they gain experience.

Some developers cited the codification of their organizations' knowledge as the major benefit of the project. By making the knowledge active in a running computer system, they were able to fully articulate their knowledge and ensure its completeness and consistency.

Conclusions and Recommendations

The generalized experience of over 150 expert system development projects suggests some heuristics for successfully managing an expert systems application:

It is unnecessary (and perhaps unwise) to undertake a massive knowledge engineering project to begin to take advantage of expert systems technology. High returns on investment can be found in automating simple knowledge-processing functions. Furthermore, these simpler systems can be built with more predictable projects, using predictable amounts of resources, and in many cases can be maintained with a very reason-able level of effort.

Starting with an isolated system will permit the novel knowledge engineering parts of the system to be created first. A stand-alone demonstration system can be created in a few work months But by the time the system is pilot-tested, expect to spend as much effort in traditional software engineering activities as you do in knowledge engineering activities.

Populate your project with people who have an application-fielding track record, not with inexperienced people who have AI degrees. It appears to be easier for experienced software developers to learn the AI they need than for the educated AI expert to acquire the practical skills needed to field a system.

Expert system projects can be managed like other software projects. Progress is generally steady, and reliable estimates of development time and effort can be created and managed, too.

If possible, develop your expert system in a hardware and software environment similar to your delivery environment. Because so much of the total effort is in system integration, it generally makes sense to develop in the environment you'll need to integrate with.

In conclusion, our sample of applications suggests very good news for organizations with a need to distribute and codify knowledge. From a management perspective, expert system development projects appear to be very similar to traditional software development projects. They can be costed, tracked, delivered, and supported in the field with the personnel, equipment, resources, and management techniques that are already in place.

Acknowledgments

The author would like to thank Alan Fisher and Donald duBain for their efforts in contacting and interviewing dozens of expert system developers. And special thanks are due to the developers themselves, who gave generously of their time and shared their hard-earned insights into the expert system development process.