End Users as Expert System Developers?

CHRISTIAN WAGNER
City University of Hong Kong

Background

Growing Need for Knowledge-Based Systems

Knowledge is gaining widespread attention as a strategic tool for the competitiveness of firms, both in the management literature and the popular press, e.g., Nonaka and Takeuchi (1995) and “In Praise of Knowledge” (Economist, May 27, 1995, p. 20). However, knowledge creation and management is not simply the capture and storage of information. It also requires the storage and processing of associations (rules) through which meaning can be derived from the information. And, to follow the argument of Nonaka and Takeuchi, knowledge should be stored in explicit, observable form.

Presently, the pre-eminent information technology vehicle for the explicit representation of knowledge in managerial applications is the expert system (or knowledge-based system). Thus, knowledge explication would require massive expert system development. Given the existing demands placed on the systems development function, the implication is that at least part of the expert system development effort will have to be completed by end users. This creates a dilemma. On one hand, the bottleneck in knowledge acquisition is well acknowledged (Holsapple and Raj, 1994). On the other hand, one of the truisms of expert systems development is that the domain expert should not be his or her own knowledge engineers (Hayes-Roth et al., 1983), a heuristic based on Johnson’s paradox of expertise (Johnson, 1983).

Despite common wisdom condemning end user expert system development, there are numerous accounts of successes with this development approach. For example, DuPont (Feigenbaum et al., 1988; McNurlin, 1987) reported positive experiences with end user created solutions 10 years ago when development tools were more primitive than today. So did Eastman Kodak (Huntington, 1989), and so did the U.S. Navy (Griesser and Tubalkain, 1992). The number of applications ranged from a few tens (U.S. Navy) to more than 1,000 (DuPont), the size per application from tens of rules to several hundred. Lubrizol has users participate in expert system development by letting them build the underlying decision tables (I/S Analyzer, 34(3), 1995, pp. 12-15). In addition, several of today’s development “languages” have become much more user friendly (e.g., visual design of trees as in Exsys’ Rulebook software) and are now targeting end user...
developers rather than professional developers. World wide sales for these desktop AI tools amounted to US$ 4 million in 1995, with annual growth rates exceeding $500k ever since 1991 (Intelligent Software Strategies, XII(2), 1996, p. 4). These trends are paralleled by numerous university curricula offering courses on expert systems to their business (I) students, who are likely not going to be professional developers.

Clearly, end user expert systems development enjoys at least a successful niche presence. Yet a recent survey on end user computing sophistication (Blili et al., 1996) did not even consider to ask participants about the use of fifth generation expert system tools. Hence it seems that most companies do not even consider this approach, while those who do are successful (or at least claim success). Will therefore those companies who embrace user developed expert systems become knowledge-based firms, while the many others will not? And, what are the limits to end user developed systems? Accounts of successes, such as those of DuPont, Kodak, or the Navy, are not sufficient to help us answer this question, as they are always presented at the macro level (company totals for the number of systems developed, number of rules per system, average pay-back). They report little about the difficulties end user developers face in actually creating their systems.

Special Problems of End User Developed Expert Systems

In a recent study, Panko (1998) addressed the issue of spreadsheet errors and their potential impact on organizations. He reports a picture of unacceptably high error rates and significant bottom-line impact related to these errors. If a similar pattern emerged in expert system development by end users, it would already be concerning. However, expert system development adds several additional elements of complexity, compared to spreadsheet development, which can potentially result in even higher error rates, namely:

- Expert systems carry out qualitative (logic based) reasoning using (frequently) non-numeric symbols whose semantics can be defined by user rules. Spreadsheets are largely quantitative with well-established semantics.
- Expert system developers will usually have little training in the “language” of expert systems, which might be a derivative of first order logic, or a similar logic-oriented language. In contrast, spreadsheet developers can rely on many years of training and expertise in the common “language” of spreadsheets, arithmetic.
- Most expert system shells separate development from use. That is, the knowledge base will be developed in an editor or other development environment (with limited error checking). After completion of the knowledge base, it will then be executed. This separation of development and use makes verification and correction of errors significantly more tedious, if not difficult than in spreadsheets, which are more WYSIWYG in nature and which provide instantaneous feedback on the correctness of a formula, or on the impact of a number change.
- Rule-based expert systems frequently adopt a backward reasoning approach whose logic is not at all obvious to inexperienced developers, or to developers with experience in traditional programming languages.

Taken together, these characteristics add extra complexity which affects knowledge extraction and formalization (extraction and formalization of “rules” or such), representation (especially in a backward reasoning environment), validation (in the absence of WYSIWYG and instantaneous confirmation), and learning/usage (logic with user definable semantics versus algebra). Consequently, higher error levels are to be expected than for more typical end user applications such as spreadsheets, and special attention needs to be drawn to key sources of errors.

Overview of the Study

Given the apparent difficulty of end user expert system development on one hand, and the increased demand for end user developed expert systems on the other hand, the decision was made to investigate the feasibility of this form of knowledge base creation. Two questions were addressed. First, are end users able to develop expert systems that are structurally of sufficient quality to represent the knowledge they are supposed to (knowledge formalization and representation), and secondly, are experts capable of expressing their reasoning sufficiently to create valuable knowledge bases (knowledge acquisition)?

The first question was investigated by assessing the quality and size of end user developed knowledge bases. The purpose was to determine the limits for size and reliability of end user created solutions and thus to assess how far end user expert system development can go. The second question was addressed through the review of research in the area of tacit and implicit reasoning. Its purpose was to find evidence for whether end-user developers can develop expert systems that convey useful business knowledge, regardless of design quality.

The rest of the article is organized as follows. First, an empirical analysis of end user developed applications is introduced, together with metrics for expert system quality. This is followed by a more detailed analysis of the problems encountered in end user development. The purpose of this section is to illustrate to the reader which aspects of the development process create difficulties for the inexperienced developer. Thereafter, the issue of knowledge explication is discussed. The last section extrapolates the findings to determine the limits of end user developed expert systems and discusses alternative approaches to capture the valuable knowledge of end users.