DOCUMENT RESUME

ED 301 166 IR 013 506

AUTHOR	Slawson, Dean A.; And Others
TITLE	A Qualitative Approach to the Evaluation of Expert Systems Shells.
SPONS AGENCY	Advanced Research Projects Agency (DOD), Washington, D.C.
PUB DATE	Apr 88
CONTRACT	ONR-N00014-86-K-0395
NOTE	15p.; Paper presented at the Annual Meeting of the American Educational Research Association (New Orleans, LA, April 5-9, 1988).
PUB TYPE	Reports - Research/Technical (143) -- Speeches/Conference Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	*Case Studies; Computer Software; *Data Analysis; *Data Collection; Evaluation Methods; *Expert Systems; Qualitative Research; *Research Design
IDENTIFIERS	*Expert System Shells; *Prototypes

ABSTRACT

This study explores an approach to the evaluation of expert system shells using case studies. The methodology and some of the results of an evaluation of the prototype development of an expert system using the shell "M1" are detailed, including a description of the participants and the project, the data collection process and materials, and data interpretation. A list of application characteristics by which a user of a case study library might search for relevant cases is presented. The benefits of the case study method are then discussed, including the positive effect of the presence of an outside observer on the process of developing the expert system, and a detailed description is provided of the problem setting, the shell, and the process of using it on a real expert system development problem. The pitfalls of the case study method are also noted, including the failure to generate useful quantitative metrics or to clearly identify any one shell attribute as the cause of the outcomes observed. A discussion of directions for future work concludes the report. (10 references) (EW)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED301166

U S DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
[✓] This document has been reproduced as received from the person or organization originating it
☐ Minor changes have been ma to improve reproduction quality
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

A Qualitative Approach to the Evaluation of Expert System Shells*

Dean A. Slawson and John Novak
UCLA Center for Technology Assessment

Ronald K. Hambleton
University of Mass. at Amherst

April 9, 1988

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Dean Slawson

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)"

*Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA. April 5-9.

The research described in this report was sponsored by Defense Advanced Research Projects Agency, Contract No. ONR N00014-86-K-0395. Views expressed herein do not necessarily reflect the opinions or policies of the sponsor.

TR 013 506

This study explores an approach to the evaluation of expert system shells--software tools that support the development of expert systems. Expert system shells are used to develop and run "intelligent" software for solving problems in education, medicine, business, and other domains. Evaluation of such software tools has proven difficult in the past. Early approaches to the problem may be classified as descriptive or functional (Novak & Slawson, 1987). An example of the descriptive approach is the use of catalogs of shells and their features (e.g. Liebhaber, 1987). This approach is unlikely to be of much utility for many potential tool users since in many cases the potential user is unaware of what features to look for in a tool or how those features will interact with the development problem at hand. The functional approach, on the other hand, is less well-developed (Novak & Slawson, 1987). Typically this consists of using small benchmarks or direct comparisons of shells to solve some well-specified "toy" problem (Beach, 1987; Richer, 1986; Waterman & Hayes-Roth, 1982). An alternative is the case study method, which is purported to provide insight into what performance may be expected when the tool is put to use on real problems.

In recent workshops for both tool developers and tool users at the RAND Corporation (Rothenberg, et al., 1987b) the case study method was espoused by tool developers as an attractive methodology for evaluation of shells. The enthusiasm for case studies was not shared by the tool users, however, who tended to take a more practical view. For the users, issues such as the cost involved in conducting case studies, difficulties in studying secret or proprietary projects, and the problems of obtaining and analyzing data were of paramount importance. Their objections were not founded upon any inherent weaknesses in the case study method, but rather on their perceptions of the difficulty of conducting usable case studies in a relatively unobtrusive manner.

This paper examines the process of using case studies in the evaluation of expert system shells. Potential applications are discussed first to provide a context. The methodology used in an actual case study is then summarized and discussed. Finally, some considerations for future use of this methodology are outlined. The full report including results of the case study can be found in (Slawson, Novak, & Hambleton, 1988). Additional detail on the evaluation method can be found in Slawson (1987).

Applications of Case Studies

Perhaps the most common purpose of an evaluation of expert system shells is to assist in the selection of a shell or shells for a project, series of projects, or long-term use by an organizational unit. Here it is necessary to match the capabilities of the shell to the requirements of one or more problem settings, with their associated organizational and programmatic constraints. The nature of the problems and constraints will vary depending on organizational focus and resources. Expectations of shells to be used in research will likely differ from those of shells intended to facilitate product development. Evaluation for long-term needs may use different methods or obtain different conclusions than evaluations which consider only short-term needs. A library of cases can be searched by such problem setting characteristics to find similar situations and the shells that were used to tackle them, along with process and outcome data. In this manner, a shell well-matched to the problem characteristics may be selected.

In research environments some software product may have already been selected; here the goal is to exercise the technology on a suitable task. Again,a library of case studies may be applied to assist in finding a suitable task. In this situation, cases using the same shell may be selected and examined to determine which applications were best suited to that shell, and why.

When shells themselves are being created or improved, case studies serve as a record of needed improvements and ideas for future development. Metrics based on case histories involving existing shells may be used to assess the merit of various capabilities, techniques, or practices which might be introduced or enhanced, while performance standards evolved from previous studies may be applied to the development of shell specifications. Finally, the case study can serve an important role in evaluation intended to determine whether new products have met their specifications.

Given the variety of situations and uses for a technology of shell evaluation, there appears to be value to an approach that can deal with a broad spectrum of data in appropriate ways. The case study is an especially promising way to meet such requirements. A case study can begin with a top level view of a real project with real problems, people and constraints. The rich source of data available from direct observation, interviews, and other methods makes it possible to then "zoom" in to areas of interest and to select from a variety of descriptive and quantitative metrics, or create new ones to suit the task.

Evaluation of M1 as used to Prototype a Psychometric Consultant

This section describes the methodology and some highlights of the results from an evaluation of a prototype development of an expert system using the shell M1™ on a personal computer. Data came from observations, interviews, self-report and other sources. Particular attention was paid to the ways in which qualities of the expert system shell determined the process and outcome of development. Concluding this section is a discussion of some benefits and pitfalls of the case study as a method of evaluating shells.

Participants and project. The participants included a domain (i.e. subject matter) expert, a knowledge engineer-programmer, and an observer-evaluator. The software being developed by the knowledge engineer in conjunction with the domain expert was a prototype expert system to assist in the process of designing criterion-referenced paper-and-pencil tests. The observer collected data for the evaluation before, during, and after the development effort.

The domain expert, one of us, was a nationally recognized researcher and consultant in the area of educational measurement. He was also an experienced teacher, serving as a professor in a school of education. The development and workings of expert systems were new to him, however. The knowledge engineer-programmer was a graduate student in education who had taken only one course in measurement and evaluation, thus his knowledge of the domain was limited. He had some exposure to the shell being used in the study, extensive experience with a related shell, and had developed several other expert systems in the field of education. The observer, one of us, was very familiar with the domain of educational measurement, but not highly experienced in the specific topic chosen for this project (reliability of criterion-referenced tests). He was also experienced in the knowledge engineering process.

The shell M1™, operating on an IBM-XT™ personal computer, was used in this study. The project was to develop an expert system prototype for selecting methods to determine the reliability of criterion-referenced tests. Expertise in this area is well-systematized and valuable, but not readily available. This task was selected for the test case for several reasons. For one, the prototype could be finished within a reasonable amount of time. Whether or not the actual task selected would be of use in other case studies, it is likely that case studies in other settings will also be limited in size. Another consideration in selecting the test case is that it was a consultation system and consultation is still the dominant style of interaction used in expert systems today. Since this type of problem is typical it should be easier to generalize what is learned about the methodology to a variety of settings. The moderately powerful, PC-based shell and the specific task were selected because they met objectives of a real development project accessible to the evaluators. Although evaluation requirements for other hardware and software environments may vary considerably from those considered in this case, small systems are very common and are likely candidates for an evaluation.

Data collection process and materials. Data were collected on all phases of the development process, in the form of interview notes, written observation records, videotapes, and participant journals. The sources of data are described below:

1. A pre-elicitation interview between the knowledge engineer, the observer, and the project director was conducted. This interview served as an occasion to collect background information as well as a time to generate expectations for the project. The background data included an inventory of the knowledge engineer's skills and experience with the domain, the shell, knowledge programming, and related domains. Expectations for the shell (Table 1) and the knowledge engineering process (Table 2) were used by the observer as a point of departure during later observations and interviews. The expectations served as a base-line to assist in noting deviations from expected progress, thus calling attention to possible interactions of shell capabilities and characteristics of the problem setting.

Table 1

Expectations for the Shell M1

Some Expected Strengths or Capabilities

Pattern matching capability
Structured selection easy to implement
Easy enumeration of legal values in menus
Presumption functions (like metarules)

Some Expected Limitations of M1

Restricted to monotonic reasoning
Lack of procedural language
Certainty factors increase complexity of program
Difficulty in dealing with data intensive tasks

Table 2

Expected Steps of Expert System Prototype Development

STEP 1. Knowledge engineer reviews the background text and cases for understanding.
STEP 2. Overview. Get expert's terminology for any categories or taxonomies he uses, generally identify the kinds of input and output for the consultation process.
STEP 3. Have expert explain the cases so knowledge engineer can understand the inference and consulting process that goes on during a typical consultation. This is at the level of identifying possible stages in consultation, overall sequence of question types, etc.
STEP 4. Continue to develop "strawman design." Determine basic functions for the application, knowledge and data sources, etc. Create a block diagram of consultation process.
STEP 5. Map cases onto block diagram. Do as many cases as time permits. Begin to elaborate and refine the rough model to deal with specific, sometimes special, cases.
STEP 6. Revise block design based on Step 5. Collect control rules. Finalize scope of the task.
STEP 7. Domain knowledge acquisition. Collect decision rules for knowledge base.
STEP 8. Begin coding initial version of prototype.
STEP 9. Continue knowledge acquisition, coding, testing until prototype is ready for further evaluation.

2. During the initial phases of knowledge engineering, the domain expert and knowledge engineer were observed as they interacted to do preliminary analysis and design for the test case. Later, the knowledge engineer was observed working with the shell and with the expert to build, test, and refine the system. The sessions were videotaped, and notes were taken to record observations and provide an index for the tapes.

3. Other data sources included written reports by the participants, the software itself, and post-interviews of participants. The knowledge engineer kept a

dated journal in which he recorded his impressions of the process of using the shell to build the system. All domain expert and knowledge engineer notes, drawings, or other paperwork were labeled, dated, and retained. The domain expert kept notes during the interviews on what he learned or things he wanted to consider further. Any work files or significant intermediate (e.g. "trashed") versions of software were also kept.

Between major steps of the software development process and at the end of knowledge engineering sessions, interviews were conducted to correlate the various observations, ask probing questions, and record experiences of the participants. The interview was also an occasion to suggest and discuss prototypical metrics and standards for the variables identified as being significant for the test case. The data collection followed a two-step observation-interview protocol called *stimulated recall*, described below.

To do a stimulated recall, the observer made notes during the videotaped sessions. The notes were intended to record comments, questions, and observations, as well as to serve as an index for the videotape of the process, made simultaneously. The videotape included a continuous date and time stamp and each observation in the notebook was also accompanied by a note of when the event occurred. Videotapes and notes were then used at the end of each day to guide stimulated recall and analysis of the observed events. After allowing the knowledge engineer or domain expert to discuss the project in an open-ended interview format, the observer would replay selected portions of the tape . The observer then asked questions about what was being observed on the tape. Thus the observer had a chance to verify and clarify his understanding of what was observed earlier in the day.

Following each step of development and each elicitation visit, interviews were conducted to help formulate new questions, evaluate hypotheses and plan subsequent data collection. The overall process is summarized in Table 3 below:

Table 3

Data Collection Process

Before knowledge engineering begins:

* Pre-interview to develop expectations for development process

For each day of knowledge engineering:

* Observations and videotapes

* Notes and questions on differences between observed and expected behaviors

* Open-ended interview

* Stimulated recall (using notes and video playback)

Between each development step or knowledge elicitation visit:

* Post-interview/analysis

* Formulate new expectations, hypotheses, questions

During observations, the observer was specifically looking for evidence of: 1) errors, difficulties, delays, problems, etc. which might be caused by weaknesses of the shell or by a mismatch of shell capabilities to the problem setting, and 2) knowledge engineering productivity boosts, positive participant attitudes, effective shortcuts, or improvements in quality which might be attributable to shell capabilities or features.

Approach to data interpretation. The notes and tapes were reviewed and observations catalogued. Categories for observations were developed and then were sorted so that relevant observations for specific questions were grouped together. The knowledge engineer's journal and the other reports were also carefully studied in light of the objectives of the study. The coding of observations in the original notes was reexamined after reading all the notes and forming new interpretations. Data were searched for both confirmatory and disconfirmatory evidence of intermediate interpretations.

Discussion and Interpretation

A description of the application characteristics is presented in Table 4. This is typical of the kind of information by which a user of a case study library might search for relevant cases. Examination of what helped and hindered the project, as well as interviews with participants resulted in identification of several shell capabilities that might have facilitated the process of system development in this case. These are featu.es not present in M1 that would be desirable for a similar project. Some highlights are presented in Table 5. Following this, some benefits and pitfalls of using the case study method to evaluate an expert system shell are discussed.

Table 4

Application Characteristics

TASK OR PROBLEM TYPE:	Selection
INTERACTION/INTERFACE STYLE:	Dialogue consultation via menu selection
KNOWLEDGE REPRESENTATION:	Rule-based with uncertainty
SIZE OF PROBLEM:	Small (for prototype)
INFERENCE AND CONTROL:	Forward chaining, essentially monotonic reasoning

Table 5

Some Desirable Shell Capabilities Not Found in M1

Editor features	Syntax directed editor
	Macro facility for variable names
	Run-time text modification to allow expert to modify text only
Knowledge-base interface	Multiple "views" of knowledge
	Selectable kinds of debug/trace outputs
	Selectable user dialogue options
Knowledge processing	Procedural language
	Consistency checking

Benefits and Pitfalls of Case Study Method.

A number of benefits of doing the case study became apparent during the process. Perhaps the most valuable was the additional evaluation and learning that took place with the presence of an outside observer providing feedback and asking questions. The knowledg engineer reported that he learned of new questions to consider when selecting a shell in the course of the study. He also reported that some of the observer's comments between knowledge engineering sessions brought conceptual issues to his attention sooner than they might otherwise have been, thus facilitating the process. Insight into the knowledge engineering process was gained. And ideas for improved shell design were generated. Rather than proving to be intrusive, the presence of an observer seemed to be positively motivating to the other participants (witness the more rapid coding attributed to the presence of an observer). It should be noted, however, that the observer was already well acquainted with the other participants and with their abilities and responsibilities pertaining to the project. Under less ideal conditions, an outsider might have been more of a hindrance.

Another positive outcome of the study was the detailed description of the problem setting, the shell, and the process of using it on a real expert system development problem. Only a portion of the data are presented here, however, and the selections may not anticipate the questions of future evaluators. It may be that secondary users of a case study will need direct access to the source data in order to find the information that is of most interest to them.

The evaluation cost about half again as many person-hours as the initial software prototype, so it was relatively expensive. The cost was higher than it would normally be due to the need in this project for travel to the expert's site. Under more typical conditions, the domain expert, knowledge engineer, and observer would be within close proximity over a more extended period of time. Although the additional costs of an observer and the data interpretation tasks were relatively high, the total cost for the data obtained was not unreasonable since the prototype was developed in less than six months. In this case, the study was limited to the initial prototyping stages of development, and in other cases it is also likely that evaluators will want to focus on some stage of the development lifecycle that is of particular interest. The intended uses of a shell will dictate which phases of the development cycle give rise to the most uncertainty and hence the greatest need for detailed evaluation. In the prototyping context, knowledge engineering and initial coding were salient. In a system that will require long term maintenance and updating, such factors as truth maintenance and extensibility become the foci of attention, hence data collection could emphasize later phases of the development effort.

One of the biggest disappointments with the study was the failure to generate useful quantitative metrics. Although the focus was intended to be on qualitative data, it was hoped that such indicators as "rate of progress" (in number of rules, person-hours, or stages of development) would provide meaningful data on shell performance (i.e. value of the shell as a knowledge acquisition aid). Unfortunately, the rate of progress in this case was so much determined by schedules and convenience that it was impossible to make any conclusions about the progress or ease of shell use in terms of the overall effort or time expended. Even if the time and effort could be controlled, it is apparent that the number of rules required for a given level of system performance will vary depending on such factors as programmer skill, as suggested earlier. It was hoped that knowing the background of the participants would assist in interpreting factors affecting quality and progress, but a way to be more precise about the "worth" of background skills when factoring out such variables was unavailable.

In spite of, or perhaps because of, the amount of data available, it is difficult to make a clear case for or against any shell attribute as a cause of the outcomes observed. It is also difficult to assign generalizable meaning to these outcomes. Nevertheless there were some important results. Significantly, the project proceeded smoothly in spite of the lack of many desirable features in the shell. An experienced knowledge engineer, especially one with strong programming skills, can reduce the impact of shell capabilities on the total project, especially if the shell allows flexibility in interfacing externals, implementing control structures, and the like. Overall, the progress of the prototype seemed to be more affected by qualities of the problem setting other than the shell than by the shell itself. With skilled participants the presence of a participant observer and the collection of summary data alone may offer most of the benefits of the case study without the need to address the more difficult aspects of interpretation and analysis.

Directions for Future Work

The primary difficulties encountered need not discourage potential users of the case study as a method of evaluating expert system shells. The more difficult problems of establishing standardized and reliable performance metrics may or may not be amenable to this approach. But the motivational benefits, project feedback, learning and descriptive data may nevertheless make the approach worth considering.

If it is possible to make more specific inferences about shell performance from case study data, one approach might be to develop a more standardized method of assessing characteristics of the project, participants and application domain. To make the effect of participant background variables easier to interpret, one might desire to use formal tests of abilities and skills. A battery of tests on knowledge engineering could be used to obtain a profile of the knowledge engineering expertise of the development team. Such instruments might consist of a criterion-referenced test measuring familiarity and proficiency with general knowledge engineering techniques, and a questionnaire which inquires about familiarity with the particular tool and formalisms being used. Differences between scores on pre- and post-development administrations of the test would also provide a metric of the educational value of the tool.

Large expert system development projects generally include several knowledge engineers, so the participant observer role could be filled without the addition of personnel. It would, however, be valuable to provide sufficient guidelines for the role of the observer so that this role becomes a skill accessible to in-house personnel. Some of the duties of the observer would be to collect and review transcripts of knowledge elicitation sessions, conduct pre-, mid-, and postelicitation interviews of participants, and apply coding schemes to data collected in the course of the study. Once appropriate methodology has been identified for the collection and analysis of data it is possible to design appropriate training. In conclusion, much can be gained from the qualitative case study approach to the evaluation of expert system shells. More standardized approaches to the functional evaluation of shells remain to be explored.

References

Beach, S. (1986). A comparison of large expert system building tools. *The Spang-Robinson Report, 2(10)*, 1-8.

Kaisler, S. (1987). Expert system metrics. Information science and technology office. Arlington, VA: Defense Advanced Research Projects Agency, 114-120.

Liebhaber, Michael. (1987). A survey of expert system development tools. *Contract No. MDA903-86-C-0210.* Fort Leavenworth, KS: U.S. Army Research Institute for the Behavioral Sciences.

Novak, J. & Slawson, D. (1987). A review on the evaluation of expert system shells. Los Angeles: UCLA Center for the Study of Evaluation (unpublished manuscript).

Richer, M. (1986). An evaluation of expert system development tools. *Expert Systems, 3(3)*, 166-181.

Rothenberg, J., Paul, J., Kameny, I., Kipps, J., and Swenson, M. (1987a). Evaluating expert system tools: A framework and methodology. *Contract No. MDA903-85-C-0030, R-3542-DARPA*, Santa Monica, CA: RAND.

Rothenberg, J., Paul, J., Kameny, I., Kipps, J., and Swenson, M. (1987a). Evaluating expert system tools: A framework and methodology--Wo‹kshops. *Contract No. MDA903-85-C-0030, RAND Note N-2603-DARPA*, Santa Monica, CA: RAND.

Slawson, D. (1987). Methods and metrics for the evaluation of expert system shells. Los Angeles: *UCLA Center for the Study of Evaluation* (unpublished manuscript).

Slawson, D., Novak, J., & Hambleton, R. (1988). Case study evaluation of expert system shells. Los Angeles: UCLA Center for the Study of Evaluation.

Waterman, D.A. & Hayes-Roth, F. (1982). An investigation of tools for building expert systems. *RAND Report R-2828-NSF*. Santa Monica, CA: RAND.