A COMPARISON OF THE PROCESS OF KNOWLEDGE ELICITATION WITH THAT OF INFORMATION REQUIREMENTS DETERMINATION

by

Jon A. Turner

A COMPARISON OF THE PROCESS OF KNOWLEDGE ELICITATION WITH THAT OF INFORMATION REQUIREMENTS DETERMINATION

by

Jon A. Turner
Leonard N. Stern School of Business
Information Systems Department
New York University
90 Trinity Place
New York, New York 10006

January 23, 1990

Center for Research on Information Systems
Information Systems Department
Leonard N. Stern School of Business
New York University

Working Paper Series

STERN IS-90-2

Acknowledgments

The helpful comments of Henry Lucas, Jr., Sue Conger, Ken Laudon, and Rob Weitz are gratefully acknowledged.

Prepared for the workshop, "Systems Analysis and Design: A Research Strategy" sponsored by Georgia State University, October 31 - November 1, 1988. To appear in: *Systems Analysis and Design*, W.W. Cotterman and J. Senns, eds., John Wiley and Sons.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-90-02

Abstract

It is argued that the process of knowledge elicitation differs substantially from that of traditional systems analysis. These differences are identified and described. The implication of this observation is that significant retraining of information systems professionals and reorientation of management will be required if knowledge based systems are to be used extensively in business organizations.

Introduction

Over the past several years, one of the "hot" themes in information systems (IS) practice has been "knowledge based systems" (KBS) or "expert systems" (ES)[1] as they are more commonly known. By certain accounts there are over 100 expert system packages[2] on the market and the trade press is rife with examples of commercial applications being developed from them, sometimes with the intent of gaining a competitive advantage. Often, because of their proprietary nature, firms are reluctant to describe these systems, in detail, or discuss their experience in building them[3].

Writing in the Sloan Management Review, one practitioner and two professors observe:

> There are considerable benefits in capturing the expert's experience and making it available to those in an organization who are less knowledgeable about the subject in question. As organizations and their problems become more complex, management can benefit from initiating prototype ES and ESS[4]. However, the questions now facing managers are when and how to start. ([Luconi 86], p. 11)

It is clear that managers are being advised to seriously consider KBS. Consequently, it is important for researchers and educators concerned with information systems to understand the implications of widespread adoption.

What are KBS? Rather than embodying powerful general problem solving mechanisms, as had been the goal of early Artificial Intelligence (AI) research, KBS contain (encode or represent) domain specific "knowledge" and mechanisms for manipulating that knowledge in order to produce systems that perform at the level of a human expert[5]. A key assumption of the KBS approach is that knowledge can be elicited from experts in a practical manner.

My thesis, simply, is that the process of "knowledge elicitation"[6] (KE) differs radically from that of conventional information requirements determination as commonly practiced by systems analysts. It follows then, if firms desire to take advantage of KBS technology, the existing cadre of systems professionals, or at least those that will be designing and maintaining KBS, must be extensively retrained. This has serious implications for professionals and managers.

---

[1]"Expert systems" connotes both 1) performance at the level of an expert, and 2) that the system contains expertise. Because of potential confusion in meaning, the term "knowledge based systems" will be used to emphasize the second meaning. However, when external references make use of, "expert systems," it will be carried through into this paper.

[2]Or "shells" as they are called, since they are configured for a particular application.

[3]Exceptions are XCON, a system developed to verify the configuration of VAX computers; the Dip Meter Adviser, one that interprets oil well log data; and MYCIN, which diagnoses and provides therapy for infectious disease which are fully described in the literature. Also, Cooper & Lybrand's EXPERTAX and Peat Marwick's LOANPROBE are additional examples of systems that have been published extensively.

[4]Expert Support Systems (i.e., systems that assist humans in decision making).

[5]Another description of KBS is that they "1) use specialized knowledge about a particular problem area (such as geological analysis or computer configuration) rather than just general purpose knowledge that would apply to all problems, 2) use symbolic (and often qualitative) reasoning rather than just numerical calculations, and 3) perform at a level of competence that is better than nonexpert humans" [Luconi 86], p. 4.

[6]Knowledge elicitation, the extraction of knowledge from humans is considered a subtask of "knowledge acquisition," the extraction of knowledge from any source. Knowledge acquisition is part of the process of "knowledge engineering" which consists of elicitation, representation, and implementation.

First, KBS are compared with conventional data processing systems (DPS), including decision support systems (DSS), in order to make the distinction between them explicit. Conventional methods for information requirements determination will then be described followed by methods for knowledge elicitation (KE). These two processes will then be compared along a number of dimensions and implications of these differences assessed for practice and research.

A Comparison of Knowledge Based with Conventional Systems

In order to talk intelligently about the process of KE, it is first necessary to have a clear idea of what KBS are and how they differ from conventional systems. Let me first assume readers have some familiarity with KBS[7].

One way to gain an understanding of ES is by analogy with conventional systems - especially with regard to differences. KBS have as their objective the support or simulation of human expertise using knowledge contributed by experts. One might say that DSS have also the same goal of augmenting human expertise. The difference between them is the way they go about it. DSS represent knowledge as a set of equations that explicitly relate factors (for example, a minimum order quantity inventory model) and as a collection of structured, homogeneous data (for example, an inventory data base). They also may provide an environment for the construction of equations and data bases, the extraction of data through queries and statistics, and methods of exploring alternatives or assumptions thereby encouraging a user to learn about the decision situation.

KBS, on the other hand, represent knowledge in a more unstructured, heterogeneous manner, referred to as a knowledge base (even though entries in a knowledge base may follow the same general format). KBS provide means of manipulating this knowledge base through a process of inferencing. An implication of the KBS approach is that knowledge is a commodity to be extracted from an expert by a process of knowledge elicitation and represented formally in a system.

But the differences between KBS and conventional systems, be they DPS or DSS, go deeper than just their data structures, the technology used to build and run them, and the human activities they are designed to support. Four distinctions are important[8].

1. **Control Regime**. Conventional systems are organized hierarchically in that the execution of subroutines follows a sequence according to predetermined branching logic. In this scheme, the name of a subroutine is just a token devoid of any semantics. In contrast, KBS are data rather than instruction driven and no explicit ordering among the subroutines is implied or assumed. An operator (subroutine) is selected based on an evaluation of the immediate problem space. In this scheme, the operator name has meaning.
2. **Depth of Reasoning**. Conventional systems encode situation - action logic. That is, when certain variables are recognized to contain particular values, a predetermined action is invoked, but there is no information contained in the system about why that action was selected. KBS, in contrast, have a goal of representing (some might even say "understanding") a situation at a level of detail that allows a meaningful dialogue between it

---

[7]If not, Davis' description of MYCIN [Davis 77] is one of the easiest to understand and Luconi provides a good management overview [Luconi 86].

[8]This section is based on work I have done with Vasant Dhar and presented in detail elsewhere [Dhar 88].

and an intelligent entity (human), i.e., where some causal elements of the situation are represented and can be used in reasoning. The extent to which this goal is attained is, partially, a function of the knowledge in a system and the way it is structured (represented).

3. **Data/Knowledge Structures and Interpretation**. Conventional systems operate on uniform types of data using relatively simple structures (i.e., fields and records; sequential and indexed). Processing of this data is control intensive and the logical rules for manipulating it are embedded in the program's procedure. KBS, on the other hand, attempt to separate structural knowledge from the component which interprets it.
4. **Noise/Uncertainty Handling Strategies**. Conventional systems recognize and reject noise or incomplete data as errors through a process of validation. KBS, however, assume that noise is inherent in the task environment. Consequently, input data can be conflicting (in that they lead to contradictory conclusions), incomplete, or different, in terms of the credibility assigned to them. As a result, KBS tend to be somewhat more resilient than systems in which explicit logic is needed to deal with each type of error.

The level at which a KBS reasons is a function of the type of representation scheme used. For the most part, rule based representations have been used to capture heuristic situation-action knowledge. Such heuristics have imbedded in them assumptions that are seldom made explicit. For this reason, if reasoning based on first principles is required, or implicit assumptions need be questioned, other forms of representation become more appropriate. The form of representation is not chosen a priori, but emerges through a process that resembles hypothesis testing and theory formulation (as the knowledge engineer learns about the domain). An implication of this is that system development is not one involving simple accretion of facts to be expressed within some chosen representation scheme, but one where the structure of the system evolves over time. Another implication is that selection of the method of implementation (i.e., a particular shell, or language) can not be made independent of the problem domain, or how the problem is to be represented. Thus, unlike conventional systems where a methodology can be learned and then transferred to other projects, each KBS may require the developers to learn a new implementation environment.

From this short comparison it is clear that a whole range of different design skills and processes are involved in building KBS as contrasted with conventional systems. They have different control structures and involve different strategies for construction. They involve different tools. For KBS, the key process is knowledge elicitation and the key design decision is the representation to be used.

In order to focus more sharply on the building process, a brief discussion of conventional methods of systems analysis is provided next.

Information Requirements Determination

The central activity in conventional systems analysis is information requirements determination (IRD) whereby the requirements for a system are established. There are three basic approaches to IRD [Davis 82]:

1. Derive them from the utilizing system, that is, from an analysis of the needs of those who will use the system. This involves studying the work that users actually perform using interviews, observations, sample documents, surveys, simulations, etc.
2. Derive them from an existing system. Either one that was previously installed or developed, another firm's system, or a purchased package. In other words, an understanding of system requirements is obtained by reverse engineering.

3. Evolve them through the process of prototyping. That is, by iterating through building - use - feedback - modification of requirements - building cycles of systems development. In this case, the system is the requirement, although it evolves in the minds of the user and system builder as iteration progresses.

All three approaches rely on asking users for information, although (1) above is most heavily dependent. Two factors limit the effectiveness of asking for information:

1. Cognitive limitations on humans as information processors. For example, because short term memory is limited, holding only five to seven chunks of information, the number of objects that can be tracked simultaneously is small. People have been shown to have a poor intuitive grasp of probabilities and statistics, possibly because of computational limitations. These constraints apply to both designers and providers of information, so there is ample opportunity for omission and distortion during conversations.

2. Biases introduced either unconsciously or purposely. People tend to be guided by heuristics in decision making, for example, using information because it is convenient or available, assessing a situation on the basis of its similarity to other occurrences, and reaching a conclusion through a process of anchoring and adjustment [Tversky 74]. These tendencies result in biases that range from ineffective search sets, to seeking confirming evidence for what people believe to be true and excluding disconfirming information.

Several other issues complicate the gathering of information by asking for it. Individuals tend to have different perspectives and often different perceptions of what is going on in a situation. Further, few people possess complete information; which makes the process akin to puzzle solving. Finally, there are often political reasons to distort information or misrepresent situations [Keen 81][9].

Two underlying (and complementary) strategies and design methodologies are used most often to guide the process of IRD:

1. **Data flow**. The primary focus is on data and its flow through various business processes. Data flow analysis studies the transformation of data in each business activity and documents this in diagrams that graphically show the relation among processes that operate on data, messages (data flows) that are passed among processes, and files. A data dictionary describes the contents of messages and files. Processes are represented by structured English procedures.

2. **Decision analysis**. The primary focus is the objectives of an operation and the business decisions that are made. Information needed for decision making is identified along with transformations and alternative outcomes. Factors that affect the selection of various outcomes are also identified and represented in tables, graphically in trees, or in structured English procedures.

Several observations about the process of IRD as actually practiced:

1. IRD is the most important part of the implementation process in that it is highly leveraged, having greatest impact on later stages. Errors (missing or incorrect requirements) introduced at this point propagate through the development process resulting in incorrectly implemented modules that are difficult and inexpensive to change. Yet, IRD is the most poorly understood stage of IS development [Turner 87].

2. Both strategies, data flow and decision analysis, are weak methods with products that are hard to validate. There are few tests for completeness or consistency.

3. The assumption behind both strategies is that requirements exist and need only be

---

[9]Making it all the more important to devise a good validation scheme.

discovered by an analyst. A more realistic picture is that requirements must be revealed and evolved. They are often conflicting, and thus need be reconciled. Furthermore, requirements can not be easily separated from solution possibilities and these are influenced heavily by experience [Turner 87].

4. Both data flow and decision analysis representations are static. It is awkward to depict changes or alterations which must be explicitly introduced. For example, in data flow analysis, one goes through the laborious process of first generating a physical representation of an existing system, then using this to derive a logical representation of that system. A new logical representation is constructed incorporating desired changes and then this is reflected in a new physical representation which is actually implemented. The result is that each system is materialized at least four times and consistency must be maintained among them[10].

5. Exception processing is hard to represent. Although decision analysis permits exceptions to be shown as branches in a decision tree or as entries in a decision table, including all of them complicates greatly the description of the system. Not representing them results in an incomplete description. Since data flow approaches do not permit the control structure of a system to be represented, exceptions are difficult to show.

In summary, the analysis of conventional systems is essentially a top-down, problem definition - problem solution - problem redefinition process. A solution vocabulary is developed, based on experience, that is then used to both define the problem in greater detail and to generate particular candidate solutions which are combined into a system design.

The process of KE is now described.

Knowledge Elicitation

Two leading computer scientists summarize the state of knowledge acquisition as follows:

> *Knowledge acquisition*, or the process of identifying and formalizing rules, remains an art. No truly automatic knowledge-acquisition scheme has been devised; instead, *knowledge engineers* must collaborate with expert informants in the domain of application, training them to think about their decisions and actions in terms that can be turned into rules and knowledge bases. This process often takes a very long time (a year or more). In the end the performance of such systems has often been disappointing. Rule based expert systems are "brittle" - they tend to fail badly for problems even slightly outside their area of expertise and in unforeseen situations. The systems are not capable of detecting patently absurd conclusions, unless rules are explicitly inserted to detect them. ([Stanfill 86], p. 1216)

There are a variety of methods for eliciting knowledge from a domain expert including interviews, having experts execute constrained processing tasks, protocol analysis, declarative knowledge elicitation, automated acquisition, and prototyping[11] [Grabowski 88]. Most of these techniques involve extracting information from a domain expert, either directly or indirectly, representing this knowledge in a system that serves as the prototype, and then testing the partially constructed knowledge base (the prototype) on representative problems in order to verify that the proper logical steps are executed and the proper conclusion (goal state) is reached. When this does not occur, information is either added or changed

---

[10]Maintaining this consistency is one of the primary functions of computer assisted software engineering (CASE).

[11]These methods tend to be used in combination, with protocol analysis being a common denominator among them. Olson and Rueter provide a good overview of the various methods [Olson 87] while Kim and Courtney describe the conditions under which each method should be used [Kim 88].

within the knowledge base and the tests rerun. This process is then repeated with the knowledge engineer and the domain expert working together until closure is reached, that is, until they are reasonably confident that the knowledge base is complete and the range of situations for which the system has been designed can be processed successfully[12].

The difference between rapid prototyping of DPS and knowledge engineering stems from the nature of the construction process and the nature of knowledge being extracted from the domain expert. The building of DPS are mostly analytic in that they follow almost directly from a decomposition of the problem. For example, once it is recognized that the solution involves updating a data base, well understood processing routines can be used to perform this function. These well understood routines are then combined into a solution.

In contrast to the analytic solutions of conventional systems, designing KBS are largely synthetic. That is, their properties are derived from the interaction of their parts. Fragments of knowledge are combined by a general problem solving mechanism recursively, with the result, that as the size of the knowledge base grows, the behavior of the system, although logical in a mathematical sense, becomes difficult to predict. Adding or changing a small fragment of knowledge can alter radically the behavior of a KBS.

The process of developing a KBS differs also from a DPS in the nature of the information, or knowledge[13] it contains. In the case of KBS, the knowledge being extracted from the domain expert is, in general, more detailed and hidden than that for a DPS. This is because the levels of human thought the knowledge engineer is attempting to reach are cognitive processes involving the contents of both short and long term memory - information of which the domain expert may not be aware. In a DPS, most of the information concerns the manipulation of visible objects, such as particular types of transactions, data flows, or variables in a model. The cognitive processes used to manipulate these objects are higher level and more accessible.

An example may make this distinction clearer. Suppose one wants to design a system that produces recipes for preparing food. The DPS approach would be to create a data base of recipes from one, or several, well accepted cookbooks. Most of the design effort would go into defining the data base, figuring out what fields should be used as indices and how to encode the preparation instructions, and the user interface. The information to be gathered from a potential user would consist of how recipe information was likely to be used, and possibly whether the system should have the capability of accepting a recipe from the user and storing it for later use. The KBS approach would attempt to extract, from an expert in preparing recipes (e.g., the author of a cookbook, or a well known chef), the principles of preparing dishes - the rules, if you like, for cooking meat to produce a desired result; the rules for seasoning; the relationship between size, temperature, and cooking time; the rules for preparing a sauce; etc. One might even go further and attempt to understand the types of (chemical) transformations that take place as ingredients are mixed together to form a sauce and how this would be perceived by a person tasting it.

---

[12]Realistically, three systems are built: two prototypes and one deliverable system. The first prototype is used to explore the problem domain and demonstrate feasibility. The second prototype is used to add in depth and breath to the knowledge base. Then, a deliverable product is built that meets user interface, robustness, and performance needs of the production environment [Cupello 88].

[13]The distinction between information and knowledge is hazy. In general, knowledge is information along with additional information about how and when it is to be used. There is the connotation that knowledge in a knowledge base may be more fundamental and general than information found in a DPS data base and it could be used independently of the specific KBS.

One of the primary methods of acquiring knowledge from an expert involves verbal protocol analysis [Todd 87] combined with a hard copy log if an application system is used in performing the job. Protocol analysis is a process tracing method that attempts to discover the dynamics of problem definition, hypothesis formulation, information search, and decision phases of human problem solving [Ericsson 84]. It involves recording the spoken articulation and actions of a subject during task execution and analyzing them at a later time. The notion is that this provides access to what information the subject examines, the manipulations conducted on this information, input stimuli, and the evaluations and assessments made [Todd 87]. It may also divulge the use of information external to the immediate task, that is, retrieved from long term memory.

Protocol analysis may be either concurrent or retrospective. Concurrent protocols involve having subjects "think aloud" during actual task execution. Retrospective protocols require individuals to recall their mental processes after having performed a task. The disadvantage of retrospective protocols is that they suffer difficulties of memory distortion and an inability to recall facts that were not internalized in long term memory. An advantage of retrospective protocols is that they do not interfere with task execution. On the whole, concurrent verbalization is thought to be a less obtrusive method of collecting data on task execution than are retrospective protocols [Todd 87].

Either neutral or structured probing may be used in protocol analysis. With neutral probing protocols, the observer does not intervene in any manner unless the subject remains silent for a protracted period, usually on the order of 15 seconds, at which point the observer provides a non specific prompt. The difficulty with these protocols is the amount of unstructured data they provide and in separating the useful from the irrelevant. In structured probing protocols, the subject is asked specific questions about task execution by an observer. The advantage of structured probing is that it results in a precise and compact protocol, more suitable for analysis and more comparable across subjects [Todd 87]. The disadvantage is that it invites subjects to consider information or procedures with which they would normally not be concerned. As such, it may depart from methods normally used. Of these alternatives, concurrent neutral protocols are thought to be the most reliable and valid method for divulging thought processes [Ericsson 84].

There are four methods of analyzing the resulting think aloud protocols that differ in the degree of task execution depth they reveal [Todd 87]:

1. **Scanning.** Examining protocols for information of interest.
2. **Scoring.** Tabulating frequencies of certain key items of interest. A variant is to establish an analysis period (e.g., every 30 seconds) and noting activities going on at that point. Scoring requires an a priori coding scheme derived from the context of the situation.
3. **Global modeling.** Formulating flowcharts and algorithms that capture the process of task execution.
4. **Computer simulation.** Developing computer programs that simulate task execution behavior and reproduce the process flow reflected in the protocol (often making use of a knowledge base constructed from scanning or scoring).

A number of criticisms have been raised about these methods of analyzing protocol data. In scanning, one attempts to identify objects of importance in the task domain, information categories and sources, heuristics and procedures, decisions, and exception conditions in order to assist in interpreting observations from the protocols. Scanning can often be accomplished by careful observation of

protocols, tested and then coupled interactively with computer simulation in order to reveal outline strategies and some of the specifics of task execution. One difficulty with scanning is that it is ad hoc.

Concern has been expressed over the objectivity of scoring, the reproducibility of coding and the amount of effort involved [Simon 79]. Some of these problems can be overcome by having two or more people score the data with the finding of high intercoder agreement, but this only increases the amount of effort required. The coding scheme is another source of difficulty. In order to insure that the results are not "data driven," it is important that the coding scheme be developed a priori based on the specific hypotheses being investigated. This can lead, however, to unanticipated events being overlooked.

Nisbett and Wilson [Nisbett 77] maintain that subjects do not have access to their higher order mental processes and consequently can not give accurate representations of what they do or why they do it. Nisbett and Wilson's study consisted of retrospective protocols so their criticism can not be extended concurrent verbalizations. The issue may be whether verbalized descriptions are accurate reflections of what subjects are doing rather than whether subjects have access to their cognitive processes [Todd 87].

Another line of criticism involves the extent to which collection of protocols is intrusive in task execution. Russo et al. [Russo 86] reviewed seven studies whose objective was to test whether concurrent verbal protocols are reactive and concluded that the only change was longer response times in some instances. Outcome measures of performance were not significantly affected. A test of whether verbal protocols affect performance in a complex managerial decision making task found no influence [Schweiger 83].

In order to verbalize cognitive processes, the outputs of intermediate operations must be available in short term memory [Todd 87]. An automatic (learned) process has no intermediate results stored in short term memory and thus the process steps can not be verbalized. To the extent that a subject makes use of learned responses in task execution, the verbal protocol will be an incomplete representation of the cognitive processes involved in the activity. This is why a trace of an expert's task execution appears less complete than that of a novice [Ericsson 84]. Consequently, should the task require extensive use of either automatic responses or long term memory, it may not be a good one for protocol analysis

The information revealed in a protocol may be dependent on the method of knowledge elicitation used (as well as on the task being performed). Grabowski found that only 30% of experts' heuristics were common, independent of the method of knowledge elicitation[14] while 70% were knowledge elicitation method dependent [Grabowski 88]. She observed that different methods revealed different aspects of the experts' knowledge. For example, scenarios revealed broad conceptual heuristics while having an expert execute actual familiar tasks revealed heuristics of an operational nature. This suggests the difficulty and complexity of obtaining reasonably complete knowledge coverage.

Finally, when an individual is involved in a task with considerable cognitive work load, he may stop verbalizing or give incomplete protocols [Todd 87]. Attempting to verbalize under these conditions will usually be reflected in a gradual decrease in performance.

In short, although concurrent verbal protocols produce a great deal of data, are difficult to interpret, may be incomplete, provide access to only what is in short term memory, and at best provide only glimpses

[14]The methods of knowledge elicitation were scenario, simulated familiar task (using a simulator), and actual familiar task.

into underlying mental processes, they are more complete representations than other alternatives.

How does the process of using verbal protocols compare with that of conventional systems analysis?

A Comparison of Knowledge Elicitation with Information Requirements Determination

It has been argued that the most problematic and important aspect of conventional systems analysis is IRD. It is during this process that basic decisions concerning a system are made: establishing the system concept, setting boundaries, deciding a strategy for allocating tasks between human and machine (division of labor), designing system structure, and identifying requirements that lead to specific detailed solutions [Turner 87].

KE is the corresponding process in building KBS. It involves a detailed analysis of protocols provided by an expert performing a representative number of familiar tasks using a variety of elicitation methods.

In order to contrast the process of IRD with that of KE, nine dimensions are considered:

1. **Objects**. These are the "things" that the knowledge engineer or analyst deals with, such as information sources, data elements, decisions, files and screen layouts.
2. **Level**. This refers to the level of abstraction at which the analyst operates.
3. **Cognitive analogy**. The basic cognitive process used by the analyst.
4. **Method**. The principal method for data gathering.
5. **Data**. What is gathered.
6. **Analysis method**. The processes used by the analyst for manipulating data collected.
7. **Skills**. The set of skills needed by the analyst.
8. **Process**. The underlying process used by the analyst.
9. **Cost**. The relative cost of accomplishing the process.
10. **Risk**. The degree of risk in achieving a successful outcome.

Note, it is not necessary for these dimensions to be complete, only that they be reasonably descriptive of system development activities. If the two processes score similarly on these dimensions, then we may say that they are similar; if not, we will conclude they are dissimilar.

Simple inspection of Table 1 suggests that the processes are indeed different. In KE the objects are, for the most part hidden. Verbal protocols and observations are used to deduce information, cognitive processes and procedures used in task execution. The objects themselves are small, heterogeneous, and possibly even external to the immediate system. That is, they may involve a fragment of non obvious knowledge held in long term memory which appears, initially, to be outside of the application domain. In IRD, the objects are fairly obvious, visible and relatively large. They can, in general, be easily materialized[15] and, thus, studied.

---

[15]An exception would be certain complex DSS models.

Dimensions	KE	IRD
Objects	Hidden	Visible
	Some external to system	All in system
	Fine grain	Course
Level	Causal structure	Surface structure
Analogy	Hypothesis generation	Outputs -to- inputs
Method	Observation	Interviews
Data	Verbal protocols	Data flows + Process
Analysis	Scoring, scanning	Validation, Leveling
	Computer simulation	Consistency checking
Skills	Investigation	Problem solving
	Experimentation	Previous solutions
	Empirical verification	Pragmatics
	Knowledge representation	System design
Process	Research	Engineering
Cost	High	Low
Risk	High	Low

Table 1: Comparison of the Process of Knowledge Elicitation (KE) with that of Information Requirements Determination (IRD)

In KE, the cognitive analogy is hypothesis generation and testing in an attempt to divulge a deeper structure of the task. That is, the knowledge engineer attempts to move below the surface representation of the task to a causal model of it. This process involves inductive reasoning. IRD uses desired outputs to identify needed inputs and then deductively figures out procedures to provide these inputs.

Data are gathered, in KE, by observation and through concurrent verbal protocols. In IRD, they are obtained through interviews which establish the needs of those who will use the system, from an existing system, or by evolving them through protyping.

In KE, scanning and scoring are used to analyze data, which are primarily verbal protocols. The consistency and adequacy of knowledge is then explored using computer simulation and heuristics deduced from protocols. Given a good scoring scheme, these methods can be quite rigorous, although they require a high degree of experience and skill. The formal model (simulation) has many of the advantages and characteristics of prototyping. IRD uses validation and leveling in data flow diagrams as the primary method of analysis. To the extent that there is an existing system, it can be reverse engineered and used as a model of possible requirements.

For KE, the skills required are those of a trained investigator. The causal structure of a process is to be discovered from fragments of its surface representation which requires patience and an inquisitive mind. Knowledge engineers need also experience with different forms of knowledge representation, formal model building, system implementation, and testing; this is the engineering portion of the task. The process is similar to that of doing applied research. IRD is a more structured process where the questions to ask and the underlying problems are better understood. Essentially, it involves mapping potential solutions on to recognized problems. The process is more completely that of engineering, where the properties of materials and the known relationships among them suggest the types of problems that can be solved with them.

Costs and risks of KE are high; it takes a long time, the work is exacting and it is not at all certain that the resulting system will operate properly. Not all problems lend themselves to a KBS solution. There are many unknowns[16] not the least of which is that few people have experience in KE. In comparison, the costs and risks of IRD are much lower, even though it is key to successful conventional systems development. Many methodologies exist to guide the design process and computer assisted software engineering (CASE) tools provides automated support for administration, project management and, to a limited extent, software engineering. But, most important, there are many people who have developed working systems, even if they can not articulate, very well, how they do it.

Conclusion

It has been argued that the process of KE is radically different from that of traditional systems analysis. The objects modeled are more detailed, the process that the knowledge engineer follows is closer to research than engineering, the skills required are different, as are the costs, risks and likely outcomes.

---

[16]For example, some of the difficulties with KBS are that knowledge in the domain may not converge; it may not be possible to represent enough common sense knowledge to build a practical system; it may not be possible to deduce a reasable causal structure, or to represent a sufficient amount of the surface structure; it may be too complex a problem, or computationally unfeasible just to mention a few potential pitfalls.

What do these observations mean for management, researchers, and systems professionals?

Businesses have shown a low tolerance for research activities outside of formal research groups. The research process does not lend itself to the tight control with which management feels comfortable. Furthermore, it is hard to assess responsibility and the payoffs are not clear even if the research is successful. This leads one to conclude that management will be slow to adapt to this new environment which will serve as a barrier to achieving some of the potential competitive advantage of KBS.

Frustration with the research nature of KBS may well lead to a forcing of outcomes. For example, management pressure on designers to select a representation because it is supported by a available shell rather than because it matches a problem. The difficulties in developing and using DSS over the past 15 years should provide some basis for reflective thought. The same management approaches that have been used with conventional systems won't work if the objective is robust KBS.

These difficulties suggest a need to change management's expectations of and approach to systems development. It has been hard enough for management to cope with the organizational change implicit in conventional systems. Not only may it be formidable to argue the cost/benefit of a KBS system, but the uncertainties surrounding the application of KE will require active management involvement.

KBS have deficiencies. For example, they don't learn, so that debugging them and making changes to the knowledge base must be explicitly performed by a human designer. KBS do not scale well. It is relatively easy to build "toy" systems and difficult to build robust ones that function well in real environments. Most managements may not have the wisdom, staying power, and discipline for this situation.

If KBS follow the trajectory of other technological innovations, practitioner learning will take place mostly through experience rather than in a classroom. This means failures before successes. There will probably be a flurry of activity followed by many failures, disillusionment and then, a few sparkling successes. But it will be difficult to generalize from these experiences. Too often management believes they don't have to understand a technology in order to apply it well. In this case, understanding may be crucial to success.

Researchers need to describe accurately the process and methods of KE and the implementation of KBS. Conceptual models of these activities have to be built and factors influencing success identified. These may well be different than the corresponding models and success factors for conventional systems. Tools and methodologies to support KE and the building of KBS have to be developed and experimentally verified. Finally, the skills required for KE (and the construction of KBS) must be identified and training programs established.

For practitioners, KE means a new mind set and bag of tools. Humans are wonderful at adapting to new and challenging situations. Undoubtedly, given reasonable management support, opportunity, and understanding, the necessary skills can be mastered and successful systems developed. This means, of course, a heavy retraining effort for currently employed systems analysts and their first line management. Everyone will not be suited to this endeavor.

The most important factor will be for all involved to adopt some of characteristics of a good researcher: restraint, inquisitiveness, judgment, and independence of thought. This process will be much more

difficult than learning a new computer language or system.

References

[Cupello 88] J. Cupello and D. Mishelevich. Managing prototype knowledgedge/expert systems projects. *Comm. of the ACM* 31(5):534-541, 1988.

[Davis 77] R. Davis. A DSS for diagnossis and therapy. *Database* 8(8):58-72, 1977.

[Davis 82] G. B. Davis. Strategies for information requirements determination. *IBM Systems Journal* 21(1):4-30, 1982.

[Dhar 88] V. Dhar and J. A. Turner. *Contrasting knowledge based with traditional systems.* Technical Report GBA 88- , New York University, Center for Research in Information Systems, 1988.

[Ericsson 84] K. A. Ericsson and H. A. Simon. *Protocol Analysis.* MIT Press, Cambridge, MA, 1984.

[Grabowski 88] M. Grabowski. Knowledge acquisition methodologies: Survey and empirical assessment. In *Proceedings of the Ninth International Conference on Information Systems*, pages 47-54. 1988.

[Keen 81] P. G. W. Keen. Information systems and organizational change. *Comm. of the ACM* 24(1):24-32, 1981.

[Kim 88] J. Kim and J. Courtney. A survey of knowledge acquisition techniques and their relevance to managerial problem domains. *Decision Support Systems* 4:269-284, 1988.

[Luconi 86] F. L. Luconi, T. W. Malone, M. S. Scott Morton. Expert systems: The next challenge for managers. *Sloan Management Review* 27(4):3-14, 1986.

[Nisbett 77] R. E. Nisbett and T. D. Wilson. Telling more than we can know: Verbal reports on mental processes. *Psychological Reports* 84(3):231-259, 1977.

[Olson 87] J. R. Olson and H. H. Rueter. *Extracting expertise from experts: Methods for knowledge acquisition.* Technical Report 13, The University of Michigan: Cognitive Science and Machine Intelligence Laboratory, 1987.

[Russo 86] J. Russo, E. Johnston and D. Stephens. *When are verbal protocols valid?.* working paper, Cornell University, Ithica, NY, 1986.

[Schweiger 83] D. M. Schweiger. Is the simultaneous verbal protocol a viable method for studying managerial problem solving and decision making. *Academy of Management Journal* 26(1):185-193, 1983.

[Simon 79] H. A. Simon. Information processing models of cognition. *Annual Review of Psychology.* Annual Reviews, Inc., Palo Alto, CA, 1979, pages 363-396.

[Stanfill 86] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Comm. of the ACM* 29(12):1213-1228, 1986.

[Todd 87] P. Todd and I. Benbasat. Process tracing methods in decision support systems research: Exploring the black box. *MIS Quarterly* 11(4):493-512, 1987.

[Turner 87] J. A. Turner. Understanding the elements of system design. *Critical Issues in Information Systems Research.* John Wiley & Sons Ltd., New York, NY, 1987.

[Tversky 74] A. Tversky and D. Kahneman. Judgement under uncertainty: Heuistics and biases. *Science* 185:1124-1131, 1974.