DOCUMENT RESUME

ED 353 969 IR 015 932

AUTHOR Dills, Charles R.; Romiszowski, Alexander
TITLE The Instructional Developer, Expert Systems, and the Front End Process.
PUB DATE 90
NOTE 24p.; Uneven quality of type may affect legibility.
PUB TYPE Information Analyses (070) -- Viewpoints (Opinion/Position Papers, Essays, etc.) (120)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Cognitive Structures; Comparative Analysis; *Educational Technology; Epistemology; *Expert Systems; Instructional Design; *Instructional Development; Man Machine Systems; Models; *Systems Development
IDENTIFIERS *Instructional Design Professionals; Knowledge Acquisition; Knowledge Representation

ABSTRACT

This paper is intended to provide the instructional technologist already possessing some understanding of expert systems with some insight into two of the many steps involved in the design and production of such systems: knowledge acquisition and knowledge structuring or representation. It is also intended to help technologists to see how they might fit themselves into the knowledge engineering (i.e., expert systems building) paradigm. A generalized model for the design and production of an expert system is examined with particular emphasis on the first two steps--knowledge acquisition and knowledge representation--and the relationship between the instructional development process and the knowledge engineering process. Characteristics of the ideal knowledge acquisition engineer are compared to those of the ideal instructional developer, showing that the two processes are very similar and that a cross-fertilization of ideas and techniques between the two fields is both possible and profitable. (Contains 20 references.) (ALF)

THE INSTRUCTIONAL DEVELOPER, EXPERT SYSTEMS, AND THE FRONT END PROCESS

CHARLES R. DILLS

ALEXANDER ROMISZOWSKI

INTRODUCTION

This paper is intended to provide the instructional technologist already possessing some understanding of expert systems with some insight into two of the many steps involved in the design and production of such systems: KNOWLEDGE ACQUISITION and KNOWLEDGE STRUCTURING (also known as KNOWLEDGE REPRESENTATION). It is also intended to help technologists to see how they might fit themselves into the knowledge engineering (expert systems building) paradigm.

We will begin by examining a generalized model for the design and production of an expert system. We will then look in more detail at the first two steps in this model, knowledge acquisition and knowledge representation (structuring). The relationship between the instructional development process and the knowledge engineering process will be emphasized. Finally, characteristics of the ideal knowledge acquisition engineer will be compared to those of the ideal instructional developer. We will show that the two processes are very similar, and that a cross-fertilization of both ideas, techniques and people between the two fields will prove both possible and profitable.

THE MODEL

The expert systems design and production model of interest to us is shown in Figure 1 (adapted from Martin and Oxman, 1988). As this figure shows, the first task involved in designing an expert system is to select an appropriate problem to solve. This is possibly the hardest of his tasks for the knowledge engineer to accomplish properly . A great many of the jobs we perform every day as instructional technologists are currently unsuitable for expert system development. For example, any task which is based upon hard-to-define knowledge, common sense, or intuition is unsuitable. So is any task in which it is unclear whether, when completed, the potential user would ever in fact use the expert system. Many other characteristics must be met by the ideal task if it is to be suitable (cf.,Martin and Oxman, 1988,p.130).

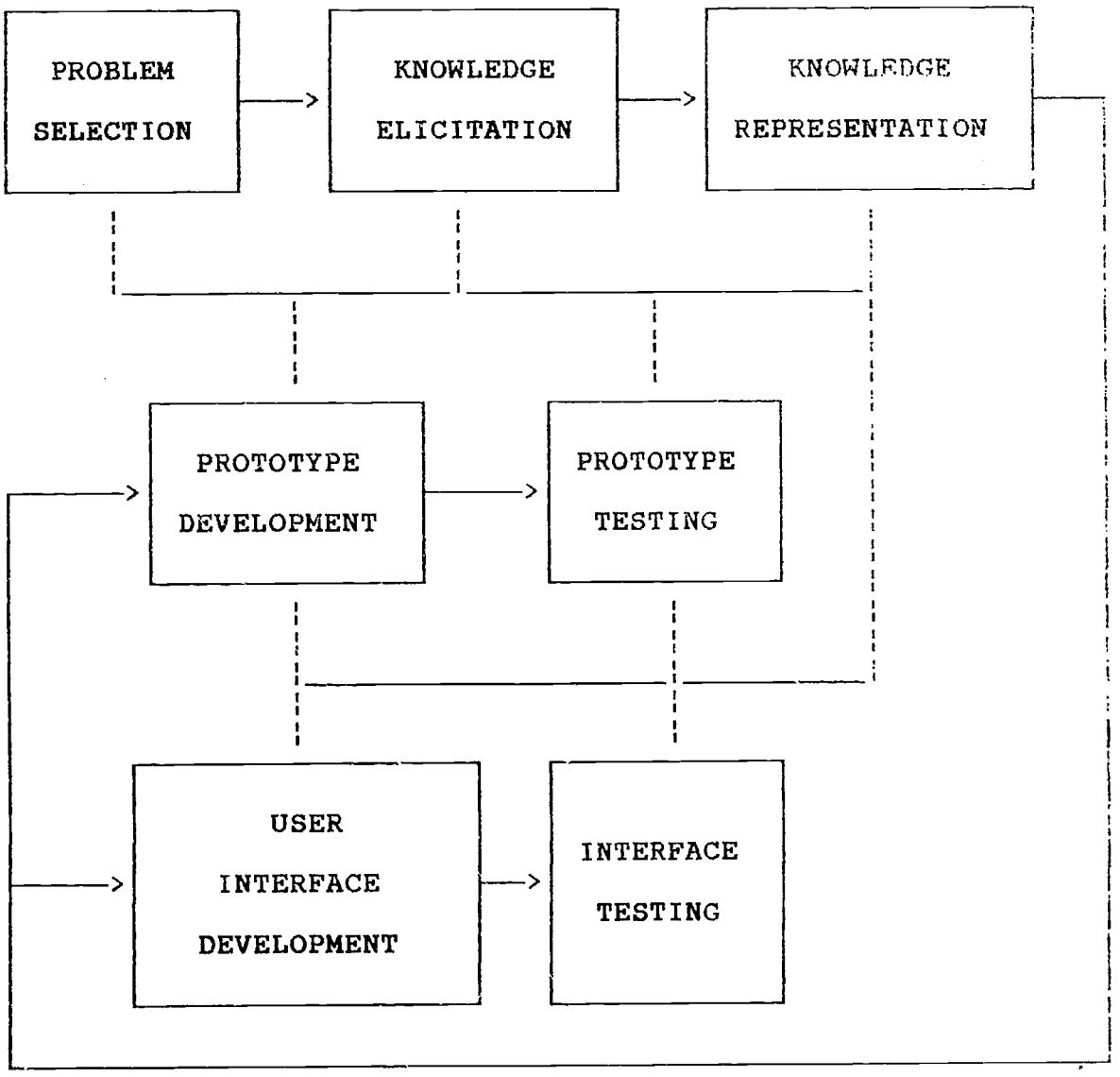

FIGURE 1 THE EXPERT SYSTEMS BUILDING MODEL

Once a problem has been selected, the next step is to obtain the data-base or set of knowledge that is the property of an expert working with this problem. There are several aspects to this step, and these will be discussed later in the paper. It is sufficient here to note that this step is probably the most time consuming and expensive step in the design and production process. It involves locating a suitable expert (called subject-matter expert, or SME, in instructional development, and a domain expert in knowledge engineering), eliciting from the expert the required knowledge, understandings, judgmental procedures and hunches, writing this knowledge in a logical and explicit form, and verifying the correctness of this written structure with the expert (Texas Instruments Teleconference on Expert Systems, 1987).

The third step of interest to us is that of knowledge structuring, or knowledge representation. This task involves converting the acquired and verified knowledge into a structure the computer can use and which is best designed to enable the user to employ the resulting computerized system. This step, also, is composed of several parts. The knowledge must be structured in such a way that the original relationships among facts, descriptions, procedures and probability spaces are maintained. It must also be structured in such a way that it can be used by the inference engine (that part of the expert system that makes decisions, inferences, and conducts other reasoning exercises). It must be structured in such a way that the factual aspects of the knowledge structure can be stored in a data base, and in such a way that efficient and convenient search and retrieval strategies can keep the inference engine supplied with whatever data it needs in order to function properly and rapidly; and it must be structured in such a way that a user interface can be constructed to the expert system enabling the user to easily ask answerable questions based upon the acquired expert knowledge. Many structuring schemes have been developed, and which one is best to use for any given problem is a factor of the knowledge domain, the type of questions to be asked by the user, the structure of the inference engine, and the expert language or shell to be used. The resemblance of this process to instructional design, media selection and other knowledge representation aspects to the instructional design process will be discussed later in this paper (Martin and Oxman, 1988; Waterman, 1986; Benchimol, Levine and Pomerol, 1987; and Taylor, 1988).

The rest of the design and production process for expert systems is much more closely related to the computer programming profession than to the instructional development profession, and so in Figure 1 all steps subsequent to knowledge representation have been lumped into one, prototyping, with one exception: prototype testing. Prototype testing is another step in the process that should sound familiar to the instructional developer. It is evaluation, including summative, formative and process, and both obtrusive and non-obtrusive, both normative and criterion-referenced, all thrown into a single pot and stirred well. Again, we will discuss this below.

One other area in which the instructional developer and the designer of expert systems can help each other is in the design and testing of the user interface. The user interface is that part of the computer system that includes the displays shown to the user, determines what inputs and other responses the user is allowed to make, and what responses the computer system can return to the user. In the case of expert systems, as with good CBI systems, the user interface also includes (and is based upon) a model of the user. In the case of the expert system, this model may be an intelligent system in itself (Systems Group, The Open University, 1987). A special language interface, known as a natural or quasi-natural language, may be employed, and this is in itself a kind of artificial intelligence system (Martin and Oxman, 1988, p.226).

This, then, is the general model of how an expert system is designed and produced. It is an overview, and much of importance has been omitted. But it should be sufficient to show those areas in which instructional developers can play a part, based upon a similarity between what they normally do and what must be done to produce an expert system. Let us now examine some of these similarities in more detail, beginning with the problem selection step.

PROBLEM SELECTION FOR EXPERT SYSTEM DEVELOPMENT

The selection of an appropriate problem for use in designing an expert system is extremely crucial to project success. However, this is not easily accomplished. Four issues are involved: identification of the task to be accomplished, identification of the people involved in producing the system, identification of the final objectives and users, and identification of both required and available resources (Benchimol, Levine and Pomeral, 1987, p. 162).

Usually a general type of problem is easily selected, in terms of either its research value or in terms of its practical value within the organization proposing the project. But the problem must be narrowed and defined, and it must be understood how the project based upon the problem might proceed. The problem to be solved must be examined in detail in order to produce a high-level design for the expert system that is going to solve the problem. The refining of the task and the definition of the expert system in terms of that task must often occur within a vacuum of knowledge about either the problem or the resulting system. This is analogous to the situation often faced in instructional development, in which the front-end, job or task analysis which, eventually, will lead to some instruction or a simulator design, must be performed before the device being simulated is yet built. Often, for example within the military procurement system, the simulator and training system must be designed long before the weapon system it supports has even been prototyped, so that the two systems, weapons and training, will reach completion at the same time. The selection of a problem for an expert system project and the conceptualization of its solution often occur within a similar atmosphere.

One area of critical importance is the selection of the expert. An expert must be available to provide the knowledge upon which the expert system is to be based, or no expert system can be produced. The first aspect of this problem is to determine the level of expertise needed. Not all (in fact, not most) expert systems are intended to function on the leading edge of a field, performing state- of-the-art work. Most are intended to aid or replace workers having a low to moderate level of expertise, or to help entry-level workers. Some are intended to perform routine aspects of a complicated multi- step procedure, or to relieve experts of routine aspects of their work. A state-of-the-art level of expertise is not needed for this sort of task. More expertise is needed on the part of the expert than

is intended to be incorporated into the program, but a world-renown expert is not needed to provide the knowledge for advising an entry-level clerk. Further, on those projects in which top-level expertise is needed, some of the demand for the expert's time can be relieved by providing a lower-level expert for use during the initial efforts. Thus, the determination of the type and level of expertise needed is very important. There seems to be no model or theory of how to do this at the present time. Some work is being done on defining the nature of expertise and of the expert (Hart, 1986). Most instructional development projects involve the identification of the subject-matter expertise needed, both in terms of quantity and level of expertise represented, as a part of the original project plan. A theoretical foundation for doing this within the instructional development world has also not been developed, but most instructional developers do this intuitively fairly well. It should prove to be a fruitful area of theory building for instructional developers.

The users of the system must be identified at this stage, and again, instructional developers have a long history of dealing with this problem, both successfully and unsuccessfully, as also have evaluation specialists. Who are the real clients for the products resulting from a project, and are they the same people who are the clients for the development process? Are they the same people who are paying for the project? Are they represented in the design process? Can the project succeed if they are ignored? Here, again, is an area in which both instructional development and knowledge engineering can benefit through the explication of the processes normally used in the instructional development and evaluation fields.

Instructional developers are familiar with the process of resource identification as a part of the initial planning phase of a project, and the same is true of the knowledge engineering project. Before the problem is finally settled upon, it must be clearly understood what resources are going to be needed, and from where they are to come. Standard resource allocation and project scheduling techniques are applicable here, but the peculiarities of the expert system development process must be kept in mind during their application (Benchimol, Levine and Pomeral, p. 164). More published case studies of this process in expert system development projects are needed, especially in detail great enough to show how standard processes can be applied, and how they must be modified in order to serve the expert systems project.

KNOWLEDGE ELICITATION: HUMAN RELATIONS ASPECTS

Knowledge elicitation and knowledge structuring or representation are often considered to be one task by writers on expert systems from the computer world. One reason for this is that they are often performed by the same people, and another is that they are usually performed simultaneously. A third reason, I suspect, is that the typical knowledge engineer knows far more about knowledge representation than about knowledge elicitation at least in a formal or structured sense.

One characteristic shared in common between the instructional developer and the knowledge engineer is the need to conserve the expert's time during knowledge elicitation. "The least amount of time the knowledge engineer requires of the expert, the cheaper the project cost will be and the faster the project will be completed (Martin and Oxman, 1988, p.434). However, very little knowledge exists concerning how to minimize the time required of the expert. The knowledge engineer should become generally familiar with the field before approaching the expert, through the reading of text books and other materials relating to the project. And he must be a good listener, quick thinker, and a rapid learner. These are also requirements for the good instructional developer. However, these are minimal requirements for optimization of the expert's time. Much more work needs to be performed concerning making optimal usage of the expert's time.

The knowledge elicitor must also maintain the expert's interest in the project, make sure he understands the role he is to play, make sure he understands the level of knowledge desired and make sure his good will towards the project and towards the knowledge elicitor is maintained. If a failure in any of these things occurs, an accurate diagnosis of the problem is also necessary if the damage is to be repaired. Again, within the realm of knowledge engineering these aspects of knowledge elicitation have not been formalized in any theory, and are usually dismissed after a short discussion summarizing their importance to project success and then dismissing them by stating that they call for the knowledge engineer to exhibit good communications and personal relations skills. While this is certainly true, it is not enough to be of any specific help in conducting a specific project.

Several lines of research conducted in instructional development can be of use in dealing with these problems as should be extended (both within the instructional development field and as applied to the expert system development process). Research on client-consultant interactions, initial meetings, appropriate models, and role changes at different points in the consulting process, have been conducted, and some efforts have been devoted to teaching students to apply the findings of this research (Rutt, 1984; Price, 1984; Coscarelli and Stoneman, 1984). Further, preliminary work by Gentry and Trimby on the formal analysis of communication interfaces for instructional development projects (Gentry and Trimbly, 1984) certainly has applicability here. Work in describing and prescribing the consultant relationship has also been conducted in other fields such as medicine and personnel psychology (i.e., Johnson, 1972, Schein, 1978; and Tilles, 1961). And the military has recently been sponsoring research concerned with the study of effective ways in which to utilize SMEs within the instructional development process. Again, these studies need to be followed up, and ways in which the knowledge elicitation process is similar to and different from the models researched in the cited studies needs to be explicated. The methodologies for pursuing and applying this research are already to be found in the instructional development literature.

KNOWLEDGE ELICITATION: METHODOLOGIES

The most commonly-discussed aspect of knowledge elicitation within the knowledge engineering literature is that dealing with formal approaches to the elicitation of knowledge from the expert. Robert Pearson (Pearson, 1988) provides a good summary of the most commonly-discussed methods. These are the structured interview, the use of probability estimation, machine induction, and the Repertory Grid. Other methods discussed in the literature include the use of historical records, observation, both obtrusive and non- obtrusive, thought experiments, and case studies (Martin and Oxman,1988, p.175).

The general methods described in these sources are for the most part those methods the instructional designer is likely familiar with and to some extent an expert in. Some exceptions might be the Repertory Grid, machine induction, and probability estimation. Pearson also discusses some aspects of the structured interview that may not be familiar within instructional development.

The Repertory Grid is a "tool used by psychologists to represent a person's view of a problem in terms of elements and constructs" (Hart, 1986, p. 174). The grid is a two-dimensional table of characteristics by cases. The characteristics are bipolar, such as strong and weak. The cases are rated on each characteristic on a scale of 1 to 5. Instructional developers have used such matrices in other contexts, such as with the semantic differential, attitude measurement, and in learning theory research. Once such a matrix has been developed by the expert, it can be analyzed for patterns, either through inspection or through more sophisticated methods, such as factor analysis, cluster analysis, or profile matching (Pearson, 1988; Hart, 1986). Further, since the world view of each expert is likely to be different from that of any other expert, it is often profitable to compare the grids of different experts.

Machine induction is a process in which the computer generates generalizations when given specific examples (called training sets) as inputs. Special software, called induction algorithms, has been developed to carry out this process. As Pearson points out (Pearson, 1988), the results are only as good as the examples chosen for inclusion in the training set.

The development of a probabilistic knowledge representation is based upon the assumption that much of an expert's knowledge is of a non-exact nature. Thus he is asked the degree of certainty he attaches to various of his decisions and recommendations. These are represented within the expert system as weights attached to other outputs, and attempt to mimic the expert's judgements in these matters (Pearson, 1988).

To use these methods of knowledge elicitation, the knowledge engineer clearly needs an in-depth understanding not only of the particular technique being used, but also of statistics and mathematics in general. Typically instructional developers and evaluation experts do not have such an extensive background in quantitative science. However, through an appropriate

re-structuring of our current doctoral programs, they could be provided with such a background. Not only would this [illegible] for performing knowledge elicitation tasks for the development of expert systems, but would allow them to use the [illegible] engineering methods within more traditional instructional development projects. What is not clear here, however, is whether a greater increase in quantitative foundations courses would discourage many of our best instructional development students from entering our programs. It may be that the cognitive style and other personality elements that [illegible] to a career in instructional development are opposed to those that would make one an expert in the mathematical [illegible] knowledge engineering. One good way to settle this matter would be to conduct a comparative analysis of appropriate personality and style elements between populations of instructional developers and of knowledge engineers. In any case, certainly some instructional developers will be found who will fit the required mold for employing these methods.

KNOWLEDGE ELICITATION: OTHER ASPECTS

In general, the problem of knowledge elicitation [illegible] the knowledge engineer is more difficult than that faced in the typical instructional development project (although there are exceptions to this) (Dills, 1988). This stems from the fact that the level and type of knowledge that the expert has must model is usually of a much more complex, more fundamental and a less well defined nature than is the knowledge represented by a CBI system or a textbook. A greater similarity exists between the two types of knowledge when the instructional developer is attempting to mold the student's attitudes, represent non-verbal behavior for the student to model or [illegible] teach in interactive process through the shaping of behavior during interactive practice sessions. It is in developing a knowledge base for these instructional activities that the instructional developer experiences the greatest difficulty and the standard methods for task analysis and content analysis are most likely to fail. And these tasks are similar to ones the knowledge engineer must face, especially in projects in which the expert system is to exhibit a high level of expertise.

The expert system must not only incorporate factual knowledge, but also must represent procedures, concepts, principles, and standards. But this is only the beginning. The expert system must reason in such a way as to mimic the expert along certain dimensions. It must sometimes make judgements which are only probably true, just as the expert sometimes does. It must make them as the expert would, in the sense that the system uses the same basis in data the expert would and [illegible] that the explanation of the decision provided to the user by the explanatory facility (see below) is one consistent with and acceptable to the expert. Finally, the judgements made by the expert system must have (at least) the same probability of being correct as the judgements made by the expert himself.

Further, the expert system usually contains an explanatory facility. The Explanation Facility tells the user, usually upon request, how a particular decision, recommendation or [illegible] was reached by the expert system. This includes displaying the reasoning pattern used, the data analyzed, and the rules utilized. Thus, the expert system must have built into it not only sufficient knowledge to duplicate the expert's output (conclusions, recommendations) but must also be able to describe the expert's thought processes and reasoning patterns [illegible] Oxman, 1988, p.28).

There are projects in which only one expert is available. It may be that the level of expertise required is such that any expert in the field would do, or that little difference among experts are unimportant at the level of the project. It may also, that only one expert is available to the project because only one expert exists at the level of expertise required. Normally, however, more than one, and in some cases several, experts are utilized. Sometimes the experts are equally utilized, as a team. More often, they are used to provide expertise about different aspects of the problem to be solved. When they are utilized to different degrees, sometimes one is the central, or main, expert, and the others are used to validate the knowledge base of the main expert, or to supplement it. Other times, the main expert is used only after the other experts have laid an appropriate groundwork, thus conserving the main expert's time. In some cases, unfortunately, the main expert is lost sometime after the project has begun, and other experts must be utilized to replace him.

In all of the multiple-expert situations listed above, the knowledge elicitation problem is complicated by the special problems involved in reconciling and validating the experts with one another. The communications and human relations problems become much more complicated because of the increased number of people involved and the greater chance for emotional problems to develop. One of the best methods currently used in such situations, by the way, is the Repertory Grid, especially when coupled with various of the structured interview techniques.

The knowledge base in the philosophy of science, the psychology of the human world-view, and the nature of the scientific method that many instructional developers and evaluation experts possess, is typically not found among professional knowledge engineers. The instructional developer can make use of this knowledge in trying to establish the reasoning patterns and paradigmatic aspects of the expert's understanding. This should enable the instructional developer to more quickly model the more fundamental aspects of the expert than would someone lacking this background (see, for example, Davies, 1984; Stahl, 1984). The use of a methodology for the generation of methodologies, such as that developed by Hutchinson and his associates (Hutchinson, 1984) for the purpose of formalizing the use of this foundational knowledge in the development of new methods for eliciting the deep structure of the expert's world-view concerning the problem at hand should prove fruitful not only to knowledge engineering and instructional development, but also to cognitive psychology and the philosophy of science.

Software for use in knowledge acquisition is being developed. One type of software, just beginning to be developed, is the Knowledge Acquisition Facility. This is a part of the software that carries on a dialogue with the expert, eliciting knowledge and procedures from him, and directly inserting them into the data base. Such a system may be specially designed for the system being constructed, or may operate through the user interface. The same system may be utilized later to update and expand the expert system as new knowledge is acquired and old knowledge becomes outdated (Martin and Oxman, 1988).

A second type of software of use to the knowledge elicitation process that tracks pieces of knowledge, the computer system and other developmental processes as well, and identifies inconsistencies within the database, as well as redundancies. One such system, called TRACE, is currently under development at IntelliSys, an expert system development company

New methods for eliciting knowledge are badly needed within knowledge engineering. Current methods are time consuming, costly, and are not guaranteed to properly elicit the needed knowledge without the intuitive intervention of an experienced knowledge elicitor. As in instructional development (Dills and Bass, 1984), much of the knowledge elicitation function in knowledge engineering is currently an art form, not a technology and so cannot be automated. Perhaps this will always be the case, but at least it should be possible to develop better techniques to use when performing the art. Instructional developers are in a good position to develop some of these, and would certainly benefit in turn from any such new techniques that might be developed.

KNOWLEDGE REPRESENTATION

Knowledge in an expert system is structured, or arranged, in two parts, which are built and stored separately and which are designed to interact with each other. One part contains the inference engine, and contains the knowledge concerned with how to reason. The other part is called the knowledge base, and contains the facts, rules and other types of knowledge used by the inference engine in reasoning.

The knowledge engineer must decide what structure to use in arranging and organizing this information so that it can be accessed and utilized by the computer, and ultimately by the user, in the most efficient and effective manner. Two questions must be answered. The first is how the inference engine is to function. The second is how the knowledge is to be organized within the knowledge base.

Both of these questions must be answered in terms of the logic of the field in question, and also in terms of the programming structures available within the computer.

Several structures for representing knowledge in the knowledge base have been developed, and are usually described in terms of how various aspects of the expert's knowledge relate to each other. Special languages have been developed to simplify the implementation of these structures, and these are likely to be used in any significantly large project. They require time to

cost for constructing the expert system, but they also [illegible] own restrictions and limitations to the resulting system. Therefore, one decision that must be made is which language is to be used to structure the knowledge. This decision must be made simultaneously with the decision concerning which logical structure to use in representing the knowledge [illegible] 1985, Chapters 3, 7 and 8; Martin and Oxman, 1988).

The instructional developer is not likely to be [illegible] with the ways in which knowledge can be structured within a knowledge base. However, he does perform tasks which are functionally similar. He structures the flow of information within lesson designs when producing CBI. If he [illegible] the CBI lesson, he structures the knowledge even more carefully. In doing this, he undoubtedly tries to discover and [illegible] natural structure to the subject matter being taught, and uses this structure in designing his lessons. He also attempts to structure the information according to design heuristics and according to what is known concerning the psychology of [illegible]. In designing a CBI or textbook page, he attempts to make use of the research on text design, and is therefore applying [illegible] logical structure to the information. In doing these things, the instructional developer is restricted by the CBI authoring language, the subject-matter structure, and the needs and abilities of the user, much as is the knowledge engineer.

Four knowledge representation schemes dominate the knowledge engineering field, and the instructional developer should be familiar with these if he is to work with expert systems. These are rules, frames, semantic networks and first-order logic. Others are being developed, and eventually may supplant these four. One should be aware of these new developments, but currently almost all work is performed using one or more of these four methods (Martin and Oxman, 1988).

Rules are the most commonly-used form. Knowledge is represented as a series of logical IF-THEN statements, and the inference engine calls these rules in a sequence determined by the subject matter, determines if the IF statement has been fulfilled, and if so, carries out the **THEN** statement. The THEN statement may be to take an action outside the computer system, to print out a decision, or to change an internal variable, such as establish the numerical value of a variable in an equation.

Frames and semantic networks are similar, and both rely upon groupings. Semantic networks consist of a series of nodes connected by lines. When drawn, a diagram representing a semantic network looks much like a flow chart for an instructional development project, or a PERT chart. The nodes represent concepts, and the connecting lines represent relationships. There is a hierarchial relationship among the concepts, such that concepts lower in the diagram inherit the characteristics of concepts higher in the structure. An example of umbrella arrangement exists. If "ship" is a high-level concept, and it is a characteristic defined into the system that "ships" have "hulls", then, if "aircraft carrier" is a lower-level concept, and is connected to the concept "ship" by the "is-one" relationship, the inference engine will automatically infer that "aircraft carriers" have "hulls" (Waterman, 1986).

Frames are similar to semantic networks. If the nodes are considered frames, then the frames indicate relationships to other frames. Each frame is a container, and the lines in the diagram connect the frames, and indicate frame-to-frame relationships. The frame itself can be considered a concept. The designer then puts facts, procedures, concept names, design information, the name of the expert who uses the concept) into the frame. This information is stored in slots. Each frame has a collection of slots, or pigeonholes, and each piece of information fills one or more slots.

Thus each frame is a cluster of related facts, and the cluster itself relates the facts to other clusters. Frames may or may not have an umbrella relationship as described above for semantic nets (Benchimol, Levine and Pomerol, 1986).

Declarative knowledge can be represented by various types of logical structures. Commonly used are propositional calculus, predicate calculus, and first-order predicate calculus. Statements such as "Lassie is a dog" and "If Lassie is a dog then Lassie is a mammal" would result in the inference engine deciding that "Lassie is a mammal" in these systems. More concerning these types of representations can be learned from Martin and Oxman, chapter 10, or Benchimol, Levine and Pomerol, chapters 3,4 and 5.

Again, most instructional developers do not have a sound background in symbolic logic, and so will have trouble [illegible] schemes. However, this situation could easily be remedied by a short course in the instructional development department or a summer institute if enough developers wanted to work with expert systems. And these schemes are easily learned, so developers can learn them on their own. If one is to serve as a front-end expert on an expert system project, one must become familiar with these schemes through one way or another.

One of the areas in which developers can aid the expert system field is to develop ways in which the internal structures of a field can be easily and quickly determined, so that information can be used in determining how the knowledge is to be represented. Again, the structure of the field can probably always be deduced from information concerning the paradigm and other world-view features of the expert's knowledge structure. As argued earlier, a formalized method for determining the paradigm of the expert and of translating it into a knowledge structure could be developed, and would be very useful. Further, most knowledge engineers have no background in this area, and many evaluators and instructional developers can

The structure of the inference engine must also be determined, and several structures (in this case, procedures for reaching a decision) are in common use. Often a ready-made inference engine is purchased, rather than custom-made, but again, its workings must be understood if an appropriate purchase is to be made. Some of the most common inference schemes are (Martin and Oxman, 1988, p.64):

-backward chaining	-problem reduction
-forward chaining	-pattern matching
-breadth-first search	-unification
-depth-first search	-event-driven control
-heuristic search	-hierarchical control

These methods of inference will not be described here, as they are fully discussed in the references. But the instructional developer must be familiar with each of them so that the proper inference technique can be matched to the problem at hand, and the knowledge base can be represented so as to properly fit the inference engine. The knowledge engineer must be familiar with the ways of building an inference engine and a knowledge base, since the way the knowledge is structured must be decided as the knowledge is being acquired

Typically an expert system is being prototyped as the knowledge structure is being developed, and early decisions are both necessary and critical to future expansions. Thus, the instructional developer, if he is to utilize his knowledge of front-end analysis in knowledge acquisition for expert systems development, must become knowledgeable concerning how knowledge is to be structured and utilized by the computer system.

Thus the instructional developer is required to develop a knowledge of inference engines and knowledge representation schemes, and this requires that he develop a knowledge of logic and of various inference schemes. It does not require that he become a computer programmer or a systems engineer. It does require that he become familiar with the languages typically used in expert system work, but on a conceptual level. In most projects, the programming will be performed by a specialist and the developer must communicate with this specialist and must know the restrictions and advantages placed upon him by the language being used.

TESTING

Not only must the resulting computer programs be tested, but the resulting knowledge structure and inference patterns must also be tested. Further, the usefulness of the resulting product to users,and the need for updated knowledge, must be evaluated. These things must not only happen when the system has been prototyped, but at each step in its evolution, and periodically after it has been completed and turned over to the customer. It is clear that instructional developers possess expertise of great value here, especially in designing and conducting user interviews, evaluating aging knowledge bases, and verifying system-made decisions against the real world.

USER INTERFACES

Instructional developers possess knowledge [illegible] two fields that are related to the design of [illegible] of expert systems. The first concerns [illegible] factors work. The design of the interface for the user is [illegible] to the design of an interface for a CBT system. [illegible] involves a great deal of screen and text design [illegible] Further, the functional design of the interface [illegible] what the user can ask the system, the type and manner in which [illegible] gives the system data, and the type of reply he can [illegible] Instructional developers already have standard ways of discovering the appropriate forms for such interactions [illegible] of the user's intentions.

A second area in which instructional developers can make a contribution to the interface design is through [illegible] of cognitive styles. It is known that not all experts think in the same manner, and that, further, experts and novices in a field think differently. The expert system must reason as the expert does, at least in high-level expert systems. [illegible] user were an expert himself, and reasoned as the expert, he probably would not be using the system at all. This is not always true, of course, but often it is. Therefore, when the novice or the other user receives an answer to a question or asks the Explanation Facility to explain how the answer was reached, it would be useful if this answer could be given in terms of how the real expert would arrive at this answer (supposedly, this is also the way the computer arrived at the answer), and also in terms of how this answer could be arrived at following the reasoning style of the user. In other words, if the expert is a "reasoner by comparisons", and arrived at the answer by comparing various factors, this solution would be explained. The computer could translate this into the user's style, which might be "reasoning by contrasts", and present the chain of inference to the user also, once it learned what his own style was.

Currently, no such system exists, but by modeling the student or other user, such a system could be developed and attached to almost any independently-developed expert system. Such a system would be particularly useful in training, where to reason according to different styles, or in using expert systems as instructional delivery devices. Instructional designers are in prime positions to design such a student model/explanation provider.

SUMMARY

It is argued that instructional developers are in an excellent position to make contributions to the field of expert systems development, particularly in the areas of knowledge acquisition, knowledge structuring, testing, and designing user interfaces.

They must become familiar with the tools, systems and approaches used in the field by knowledge engineers in order to make this contribution.

Once they have learned the fundamentals of the field, they can apply their specialized knowledge and experience both as practitioners and as researchers. In doing so, they will apply a new technology to educational use, as well as provide the knowledge engineer with new tools that he will find quite useful.

REFERENCES

1. Benchimol, Guy; Levine, Pierre; and Pomerol, Jean-Charles. Developing Expert Systems for Business. Kogan Page/North Academic Press; London, 1987.

2. Coscarelli, William and Stonewater, Jerry. "Psychological Typologies and the Dynamics of Consultant Relationships." in Instructional Development: The State of the Art, Vol. II. Edited by Ronald Bass and Charles Dills. Kendall/Hunt Publishing Co.; DuBuque, Iowa; 1984.

3. Davies, Ivor. "Instructional Development Paradigms: Archetypes, Paradigms and Models." in Instructional Development: The State of the Art, Vol. II. Edited by Ronald Bass and Charles Dills. Kendall/Hunt Publishing Co.; DuBuque, Iowa; 1984.

4. Dills, Charles. "Expertise, Knowledge Engineering and Knowledge Structuring". Paper delivered to Syracuse University School of Education, Department of Instructional Design, Development and Evaluation, Professional Development Summer Institute, Syracuse, New York; 1988.

5. Dills, Charles and Bass, Ronald."Instructional Development: Art? Science? Both?" in Instructional Development: The State of the Art, Vol. II. Edited by Ronald Bass and Charles Dills. Kendall/Hunt Publishing Co.; DuBuque, Iowa; 1984.

6. Gentry, Castelle and Trimby, Madeline. "Visual Analysis of ID Systems." in Instructional Development: The State of the Art, Vol. II. Edited by Ronald Bass and Charles Dills. Kendall/Hunt Publishing Co.; DuBuque, Iowa; 1984.

7. Hart, Anna. Knowledge Acquisition for Expert Systems. Kogan Page; London; 1986.

8. Hutchinson, Thomas. "Metamethodology." in Instructional Development: The State of the Art, Vol. II. Edited by Ronald Bass and Charles Dills. Kendall/Hunt Publishing Co.; DuBuque, Iowa; 1984.

9. Johnson, R. "Individual Styles of Decision Making: A Theoretical Model for Counseling." The Personnel and Guidance Journal. 56(9), 1978; pp. 530-536.

10. Martin, James and Oxman, Steven. Building Expert Systems: A Tutorial. Prentice Hall; Englewood Cliffs, New Jersey; 1988.

11. Pearson, Robert. "Knowledge Elicitation: A Study Guide". In Expert Systems in Education and Training. Edited by Alexander Romiszowski and Robert Young. Syracuse University School of Education, Department of Instructional Design, Development and Evaluation. Professional Development Summer Institute. Syracuse, New York; 1988.

12. Price, Robert. "The Initial Client Conference: Implications for the Continuing Relationship." In Instructional Development: The State of the Art, Vol. II. Edited by Ronald Bass and Charles Dills. Kendall/Hunt Publishing Co.; DuBuque, Iowa; 1984.

13. Rutt, David. "Consultation in Instructional Development: A First Look." in Instructional Development: The State of the Art, Vol. II. Edited by Ronald Bass and Charles Dills. Kendall/Hunt Publishing Co.; DuBuque, Iowa; 1984.

14. Schein, E. Process Consultation: Its Role in Organizational Development. Addison Wesley: Reading, Massachusetts; 1969.

15. Stahl, Robert. "Cognitive Theory Within the Context of an Information Processing Model and Learning Hierarchy: A Viable Alternative to the Behavioristic System." in Instructional Development: The State of the Art, Vol. II. Edited by Ronald Bass and Charles Dills. Kendall/Hunt Publishing Co.; DuBuque, Iowa: 1984.

16. The Systems Group, London University. Introduction to Systems. Manpower Services Commission; London, England; 1987.

17. Taylor, William. What Every Engineer Should [illegible]
AI. The MIT Press; Cambridge, Massachu[illegible].

18. Texas Instruments. "Third Annual Texas Instruments
Teleconference On Expert Systems." [illegible]

19. Tilles, S. "Understanding the Consultant's [illegible]
Harvard Business Review; 39, 1961. pp. [illegible].

20. Waterman, Donald. A Guide to Expert Systems.
Addison-Wesley Publishing Company; Reading,
Massachusetts; 1985.