XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

Knowledge Engineering: Prototyping a Small Knowledge Based System

by
Walter G. Rolandi
Knowledge Engineer
NCR Advanced Systems Development
Engineering and Manufacturing, Columbia, SC

Introduction

Although several sources have recently appeared on the subject of knowledge engineering, there is no clear consensus as to how the software developer should go about that task. In fact, with only a handful of exceptions, (for an excellent discussion, see Hoffman, 1987) most of that which has been published describes little more than a general theory or some skeletal, non-specific methodology. As more and more software engineers, programmers, and analysts, especially those coming from MIS or DP backgrounds, become involved in the development of knowledge based systems, the demand for specific descriptions of actual projects will grow.

This article seeks to address that demand. It is intended to briefly chronical the development of a knowledge based prototype at NCR Advanced Systems Development. Some of the details of the application are slightly fictionalized in order to avoid identifying particular individuals involved in the process and to protect proprietary interests. The overall scope and intent of the project is however clearly stated as are the knowledge engineering procedures undertaken to automate the process. Explanations of significant decisions and developments in the project life cycle are included.

The article picks up at a point in the project life cycle where the need for a knowledge based system has been identified, and the problem has been initially investigated. Basics as to management expectations, user requirements, and system development constraints have been determined. An "expert" has been identified and a feasibility prototype has been commissioned. The circumstances suggesting the need for a system are described.

Basic Background of the Problem

NCR customers are provided with toll-free telephone access to both hardware and software support. This support is provided on a nationwide bases. All calls, regardless of their origin in the U.S., are first channeled through a central communication office. Clerical telephone and CRT operators answer each call, obtain vital customer and product information, construct a description of the problem that the customer is having, and ultimately pass the "incident" on to an intermediary analyst. This intermediary analyst acts as an expediter and problem classifier. He scans the incident report to classify the problem as hardware related, OS related, application related, etc. and if appropriate, passes the incident on to more specialized analysts who re- establish contact with the customer and solve the problem. Part of this intermediary analyst's job is to recognize obvious hardware problems, confirm the problem by recontacting the customer, and dispatch a field engineer to repair whatever is wrong. This procedure allows a relatively fast turn around response to obvious hardware problems and also prevents the wasteful involvement of more specialized analysts.

Automation Task

The intermediary analyst and the abilities he possesses obviously represent a valuable asset. His skills help provide customers with timely resolutions to their problems while conserving highly specialized resources in order that their expertise can be more advantageously employed to solve more difficult or subtle hardware or software problems. The intermediary analyst can personally resolve as many as 11% of the total daily incidents. Because of his prowess for classifying problem descriptions and expediting problem resolutions, his "expertise" became the object of our analysis. Our goal was to build a system which would assist clerical personnel in classifying problem descriptions in a manner that was functionally equivalent to the expert. Upon completion of a successful prototype, the system would then be transparently interfaced into the existing database system used by the clerical telephone and CRT operators who answer the customers initial calls. Implementing the system as such, the customer would be better served. By virtue of the fact that the system would provide them with the knowledge to accurately assess the customer's problem, the clerical personnel could either fix it or immediately dispatch a field engineer if needed.

Getting Started

As with many software engineering projects and particularly with knowledge based systems, sometimes getting started can be a problem. Upon initial inspection, the problem domain might appear chaotic or seem overwhelmingly complicated. A large part of the knowledge engineer's job is to find order in such seemingly disordered decision making processes. The knowledge engineer cannot acquire nor functionally represent the knowledge of an expert without first obtaining a predictive understanding of what the expert does when they exercise their expertise. Behavior analytic methods are extremely effective to this end. (Rolandi, 1986).

"Predictive understanding" is meant to include answers to questions such as:

Exactly what decisions does the expert make?

What are the decision outcomes?

Which outcomes require greater reflection, exploration, or interaction than others?

What resources or inputs are required in order to make a decision?

What conditions are present when a particular outcome is decided?

At what point after exposure to decision influential inputs is a decision made?
How consistently do these conditions predict a given outcome?

and ultimately:

Given the particulars of a specific case, will the outcome predictions of the KE team be consistent with that of the expert?

Although basics obviously would precede more specifics, no other particular sequence is implied by the above listing. Answers to these questions need to be obtained as the questions themselves arise, regardless of when they arise in the knowledge engineering process. The specifics of how these questions were answered are described.

What decisions were being made by the expert?

In order to answer this question, the knowledge engineering team needed to define the scope of the overall decision making process. Initial interviews with the expert were arranged and conducted. Because behavioral analysis is an empirical process, descriptive quantitative data was sought. Since preliminary analysis of the process had identified the incident reports as a necessary component of the expert's decision making, a large sample set of incident reports was obtained for analysis. In order to identify the characteristics of a "typical" decision, an inexpensive tape recording system was put into place that allowed recordings of the conversations between customers and the expert. As mentioned earlier, these conversations occurred when the expert recontacted customers after having read their incident reports. The expert was asked to mention the incident number on the recording so that the team could later correlate that expert/customer interaction with a copy of the incident report. A series of taped interactions were transcribed and coordinated with their respective incident reports. Thus a wealth of data that realistically depicted the creation, classification, and resolution of a large set of cases was compiled.

What were the decision outcomes?

Perusing the body of data, the team was able to clearly identify decision outcomes. For the purposes of our automation task, there were basically three outcome categories: obvious hardware, obvious software or application, and other. "Other" essentially included everything (regardless of whether it was ultimately determined to be hardware or software related) that was not identified in or inferable from the incident reports. Given the team's objective to automate recognition of only obvious hardware cases, the focus of our analysis shifted to that data only.

Which outcomes required more interaction than others?

Using all the cases in our library that the expert identified as obvious hardware problems, the team compiled statistics on the number of conversational interactions between expert and customer. It was discovered that all of these interactions were characteristically short with a "typical" interaction involving only two exchanges of information. Upon inspection and classification of those interactions, it was determined that in 80% of the cases, the interactions served only to confirm the problem description as recorded on the incident report by the clerical telephone operator. This meant that whatever was necessary to make an immediate hardware decision was,

in most cases, contained in the problem description. Given that we had apparently located the decisive input for 80% of the obvious hardware cases, the problem descriptions became our next focus of analysis.

What resources or inputs were required to make a decision?

Although the team was convinced of the preeminent influence of the problem description on the decision making process, we needed to be certain that there were no other more subtle forces at work. The expert was observed in his work environment under his typical work conditions. He was asked to report the point at which he felt that he had made a decision for a number of cases. He was also asked to state what caused him to come to this decision. As expected, he reported information in the problem description alone. Additionally, he reported that his decision was often already made before he reestablished contact with the customer.

The expert was then asked to decide cases without access to the incident report. In other words, a knowledge engineer held the incident report reading aloud only the information the expert requested. In all cases, a decision was made based on the problem description alone. In retrospect, this may seem like overkill. The incident reports however contain several sources of information that could well have contributed to the expert's decision outcome. The team needed to isolate only those sources that were in fact decisive.

What conditions are present when a particular outcome is decided?

What information was extracted from the problem descriptions that accurately pointed to an obvious hardware classification? In order to address this question, the team began an in depth analysis of all of the problem descriptions that were identified as representative of obvious hardware decisions. Having previously been entered into a commercial database system, the problem descriptions were electronically available. A "C" program was written to parse the problem descriptions into a simple list of all words contained therein. The UNIX sort utility was employed to produce reports that alphabetized and listed all of the problem description words and additionally provided an indication of their relative frequencies. From these reports, the team was able to immediately identify a list of words that predicted obvious hardware decisions. The list that emerged was a list of peripheral hardware devices. Additionally, there emerged a list of words that when associated with a particular device, inevitably resulted in a hardware decision. Some of the descriptors had a "heuristic" effect. That is, the expert's decision could be predicted given the presence of certain descriptors regardless of the device that was implicated. This meant that there are attributes for all instances of computer hardware that if instantiated with a particular value, reliably predict hardware malfunction. These more global predictors emerged as a function of writing a number of problem descriptions on small pieces of paper and affixing them to a blackboard for analysis. Acting out a kind of "categorical imperative", the team arranged the problem descriptions into clusters by device type. The device type clusters were then subcategorized into problem descriptions with similarities. Not only did this exercise reveal patterns across hardware devices, it also discerned patterns of device specific problems.

How consistently did these conditions predict a given outcome?

These word lists so reliably predicted a hardware decision that the team decided to experiment with a very primitive natural language interface. This interface was designed to simply scan the problem descriptions first for words that identified a specific hardware device. Secondly, (if the first pass failed), it scanned the text for alternative words that reliably predicted or indicated the

involvement of a specific hardware device. And lastly, the program scanned the problem description for particular words that when associated with a known hardware device always indicated a field engineer was needed. It was hoped that this system would automatically flag all clear cut hardware cases and leave other possible hardware cases to be interactively examined by the system while the telephone operator extracted system prompted input requests from the customer by phone.

As it turned out, the immediate identification of "clear cut hardware cases", whether by expert or machine, was totally dependent upon a clearly stated and informative problem description. For problem descriptions that included or identified a hardware device or a part or component of a hardware device, and a predictive descriptor of that device or part, this system worked amazingly well. Initially we were worried about false positives where the problem descriptions contained the important keywords but not in the way that reliably implicated a known problem with a known device. As it turned out, this was not really a problem. This scheme's weakness was that it had to rely on the problem descriptions without benefit of subsequent interaction and unfortunately, only a relatively small number of the free form problem descriptions contained the information needed to make a decision. For the most part, they were too general ("system inop") or not actually descriptions of computer problems at all. For example, many problem descriptions were really assessments of the urgency of the solution ("checks won't print. need ASAP"), or statements referring to the emotional state of the customer ("problem with system, customer sounded frustrated"). In cases such as these, the expert would typically recontact the customer by phone to obtain a more specific description of their problem.

Although the simple parser scheme accurately identified all obvious hardware problems which had problem descriptions including a device, a device part, and a device or device part descriptor, we abandoned the scheme because it simply was too limited in its range. In other words, the percentage of the hardware cases that it would automatically flag was not significant although those cases so flagged were accurately identified. As an alternative, a menu system was implemented in a commercially available inference engine running on an NCR Tower 32.

Our experimentation with the parser was most productive however: by the time we had decided to abandon the interface, we had exhaustively identified all hardware devices and their component parts that if described by a member of a limited set of descriptors, reliably indicated that a field engineer was needed. We sought to obtain a measure of the consistency of these predictors with the decisions of the expert.

Using a set of incident reports that the expert had never before seen, the expert was asked to classify the problem as being related to hardware, software, or other. The incident reports were obtained from a history file that included a description of the ultimate solution to the problem. Thus, the team could identify which incidents involved actual hardware issues and which did not. By the way, with a large number of cases, this sort of exercise will typically reveal some degree of inconsistency in the performance of the expert. That is, cases with identical problem descriptions would occasionally be reacted to differently by the expert. This is important to note but is only problematic when the administration of the expert's expertise becomes unpredictable. Small inconsistencies are to be expected but true unpredictability would signify the absence of expertise.

The outcome of this exercise was to further reinforce our faith in the reliability of the keyword predictors. There was essentially a one-to-one correspondence between our predictions of what the expert would do given a certain problem description and what he actually would do. Needless to say, this was encouraging.

At this point in the knowledge engineering process, we had answered most of the questions one needs to answer to arrive at a predictive understanding of the expert's decision making behavior. We had discerned that the truly obvious hardware decisions would not be made by the expert unless he had first identified one or more of a limited number of devices or device parts, and one or more of a limited number of descriptors of those devices. Their presence essentially was an assertion of fact and as such, could be succinctly represented:

Device	Component or Part	Descriptor
printer	ribbon	tangled
terminal	key	jammed
tape drive	door	broken

The determination was driven in prototypical implementations by the backward chaining rule:

if	a hardware device is identified
or	a part, component, or attribute of a hardware device is identified
and	a significant descriptor of that device or component is present
then	there is an obvious hardware problem and a field engineer is called for.

The purpose of the menu system was to effectively construct an instance of some fact which was recognized as constituting an obvious hardware decision by our expert and represented in the knowledge base.

Of course not all hardware decisions were representable as constructable facts. As mentioned earlier, some problem descriptions do not contain the necessary information for an immediate classification. In such cases, the expert would telephone the customer and ask them to elaborate on their problem. This interaction would continue until such time as the expert had classified the problem as hardware or knew that it should be passed on to one of the analysts. His interaction with the customers not only provided him the means to formulate and test as many hypotheses as needed to classify the problem, it occasionally provided an opportunity to solve some of the customers' problems. This is because the expert responded to cues in the follow up customer interactions that could indicate any of a number of more subtle kinds of problems. The ability to make these sorts of discriminations was sought for the system as well.

This aspect of the KE process entailed a slightly different approach. We had already obtained an accurate assessment of the scope of the domain, that is, the range of things about which something had to be known. We had also discerned the critical components (as well as an effective way to functionally represent them) of the more common sense, obvious hardware problems. These types of problems essentially entailed a one-to-one mapping from something contained in a problem description to some actionable fact contained in the knowledge base. But what of less specific problem descriptions or subsequent conversational interactions? What was it about these "inputs" that served as a basis for action? Another way of asking this question is, "What does the expert have to *otherwise* know in order to respond to the cues he detects in conversation with the customers?"

What knowledge is subsumed by the assertion of given facts?

The expert was asked to role play a number of follow up scenarios as if he had been given incident reports containing very non-specific problem descriptions. It was soon discovered that, given a particular topic of discussion, (either a class of device or a class of problem indicators), he would seek to determine answers to only certain kinds of questions. For example, if some unspecified thing was wrong with the customer's printer, the expert would typically ask the model number of the printer. His subsequent conclusions or line of questioning were totally contingent upon the response of the customer. This is because knowing only one fact (the model number) actually implied knowing a great number of facts. It was almost as if upon learning a model number, the expert "loaded in to memory" as many as fifty other pieces of information that are associated with that fact. Regardless of the psychological mechanism actually entailed in this process, the analogy served the interest of the

project well.

General rules of inference that functionally conformed to the interactions of the expert were encoded in the knowledge base. External files were employed however to tabularly contain the large number of facts about printers, terminals, modems, etc.. These facts were then "loaded in" as a sort of frame of reference when called into play by the inference mechanism. What this netted was the ability to make very refined and specific decisions based upon relatively sparse inputs. Knowing what he knew about a wide range of hardware objects, the expert was capable of responding to a number of rather subtle cues that, to him, implicated certain kinds of problems or certain kinds of problem solutions. By determining a substantial subset of what that knowledge entailed, we were able to realistically emulate his problem solving behavior even with more obscure problems.

This ability has proven to be quite impressive. First of all, the representation scheme is straightforward, easy to maintain, and capable of providing logical branches to a vast number of problems. It also provides a means to appropriately react to and solve a number of problems that it discerns. It is this ability that seems to convey the impression of "intelligence" when the prototype is demonstrated. Reactions to this latest prototype have been quite favorable with viewers often being impressed at the range of what the system "knows".

Summary

The knowledge engineering process employed to construct a working prototype of a knowledge based classification system has been described. The approach taken was exploratational and experimental in nature. Quantitative techniques borrowed from behavior analysis were advantageously employed. The process entailed the construction of several prototypical implementations before the current knowledge representation scheme was adopted. The events described represent about 7.5 man-months of data collection, data analysis, and prototype programming. It is hoped that the specific methods and procedures described will be helpful to other software engineers involved in the development of knowledge based systems.

References

Hoffman, R. R., (1987), *The Problem of Extracting the Knowledge of Experts from the Perspective of Experimental Psychology*, AI Magazine, Vol. 8, No. 2.

Rolandi, W.G., (1986) *Knowledge Engineering in Practice*, AI/Expert Magazine, Vol. 1, No. 4.

Bio-line

Walter G. Rolandi is a knowledge engineer with Advanced Systems Development, NCR-Columbia. Interested in machine learning and language acquisition, he is additionally pursuing graduate training in the departments of psychology and linguistics at the University of South Carolina.

Acknowledgements

Particular appreciation is extended to Diana Austin and Ken Steimer for both their technical expertise and their insights into the nature of the problem. Additionally, I would like to thank our management, Jim Davis and Wayne Miller for their support and encouragement. Lastly, I would like to thank our ultimate expert, Mr. Jerry Hodges.