S19-82
34079
P-7

N95-19644

TARGET'S ROLE IN KNOWLEDGE ACQUISITION, ENGINEERING, VALIDATION, AND DOCUMENTATION

Keith R. Levi, Ph.D.
Department of Computer Science
Maharishi International University
1000 North Fourth Street, FB1044
Fairfield, Iowa 52557
email: levi@miu.edu

ABSTRACT

We investigated the use of the TARGET task analysis tool for use in the development of rule-based expert systems. We found TARGET to be very helpful in the knowledge acquisition process. It enabled us to perform knowledge acquisition with one knowledge engineer rather than two. In addition, it improved communication between the domain expert and knowledge engineer. We also found it to be useful for both the rule development and refinement phases of the knowledge engineering process. Using the network in these phases required us to develop guidelines that enabled us to easily translate the network into production rules. A significant requirement for TARGET remaining useful throughout the knowledge engineering process was the need to carefully maintain consistency between the network and the rule representations. Maintaining consistency not only benefited the knowledge engineering process, but also had significant payoffs in the areas of validation of the expert system and documentation of the knowledge in the system.

INTRODUCTION

Developing an expert system involves processes of knowledge acquisition, creation of a prototype rule base, creating a series of refinements to the prototype system, validating the system, and maintaining the system. Although these processes are conceptually separable, in actual practice they interact a great deal. The knowledge acquisition phase of this development process has long been recognized as the greatest challenge to building an expert system (Hayes-Roth, Waterman and Lenat, 1983).

The TARGET (NASA, 1993) task analysis tool appears to be well-suited to assisting in the knowledge acquisition process in a way that should directly facilitate all the other phases of expert system creation. TARGET (Task Analysis Report Generation Tool) is a graphical network creation tool intended for modeling tasks that are primarily procedural in nature. An example of a TARGET generated network is shown in Figure 1.

TARGET has mainly been used for knowledge acquisition, both as the initial phase of expert system development and also as a self-contained model of a given procedure. It was our expectation that in addition to facilitating knowledge acquisition for an expert system, TARGET would benefit the other processes of expert system development. First, it should benefit the initial creation of rules by acquiring knowledge in a form that is easily translatable into rules. Second, having a graphical representation of knowledge should assist in examining the rule base for areas needing further development and refinement. Third, the graphical representation should also facilitate validation by providing an easily navigable map of the entire knowledge base. And finally, it should aid system maintenance by providing documentation of the knowledge in the expert system in a manner complementary to that provided by the rule base.

APPLICATION DOMAIN

The application domain was a help desk for a commercial software product. Expert systems are becoming quite successful in this domain (Rewari, 1993). Perhaps the most prominent example is Compaq Computer, who have developed and fielded systems that recently received on of the American Association of Artificial Intelligence's awards for innovative applications of AI (Hedberg, 1993). Compaq estimates that their system saves them $10 - $20 million dollars annually in customer service costs. Hewlett-Packard, Digital Equipment Corporation, and Color Tile are other examples of companies that have developed help-desk expert systems (Arnold, 1993). Color Tile's system has reportedly enabled them to expand the services of their help desk, reduce average call time from ten to two minutes, and simultaneously reduce staff from twelve to seven people.

The software product for our expert system, ClearAccess (ClearAccess Corporation, 1993), provides a common end-user interface to over 60 databases, including Oracle, Sybase, Ingres, DB2, IMS, Paradox, DBASE, and others. The system works in client-server architectures, allows database queries to be constructed using a graphical interface, and also interfaces with spreadsheet, wordprocessor, and graphics programs. It can be used in conjunction with a spreadsheet program such as Microsoft Excel to create a spreadsheet front-end to a database.

Technical support for ClearAccess is a signficant challenge for its help desk. In typical software companies, technical support personnel are often junior programmers or even non-programmers. There is high turnover since personnel are often promoted to development positions when they become more knowledgeable about the product. Technical support for ClearAccess is particularly challenging because of the breadth of knowledge required. Technical support personnel must be knowledgeable not only about their own product, but also about the many databases, spreadsheets, and other packages their system interacts with. Since ClearAccess usually works in a client-server network they must also be familiar with client-server and networking software and hardware.

The help-desk expert system we developed for ClearAccess presently encompasses about 200 rules and covers the most critical and common issues encountered by the technical support personnel. The expert system is a rule-based data-driven reasoning system for diagnosis, testing, and remediation of customer problems. The system is presently undergoing validation testing by the help-desk personnel.

KNOWLEDGE ACQUISITION

The first stage in developing an expert system is the process of knowledge acquisition. This process traditionally involves two knowledge engineers interviewing a subject matter expert. One knowledge engineer has lead responsibility for conducting the interview. The other has lead responsibility for taking notes during the interview. In contrast to the traditional knowledge acquisition process we found that only a single knowledge engineer was needed when using TARGET.

During a knowledge acquisition session the domain expert articulated tasks and procedures used by ClearAccess help-desk personnel for a given problem. As each task was articulated the knowledge engineer immediatly recorded these as nodes and links in a TARGET task network. The domain expert observed the network as the knowledge engineer entered it. The knowledge that was being communicated (or not communicated, in some cases) was immediatly obvious. In addition to facilitating knowledge acquisition by making it immediatly obvious what was being communicated, this procedure also provided an explicit visual trace of the knowledge in the system as the session proceeded. This was valuable because it allowed the domain expert to

easily recall any earlier assumptions that she or he made in the course of a long line of reasoning.

Figure 1 shows a fragment of the TARGET task network for problems associated with linking databases with spreadsheets such as Microsoft's Excel. One traces a problem solving session starting with the leftmost node and the following a path according to the comments in the nodes and the labels on the directed links between nodes. The hexagon nodes are decision nodes. There are always two or more arcs leaving a decision node. The text in the decision node poses some question. The arcs leaving the node are labeled with the possible answers to the question. The rectangular nodes are required tasks. They describe a task or procedure that must be performed. Required tasks with heavy borders are terminator tasks. They indicate the completion of a process.

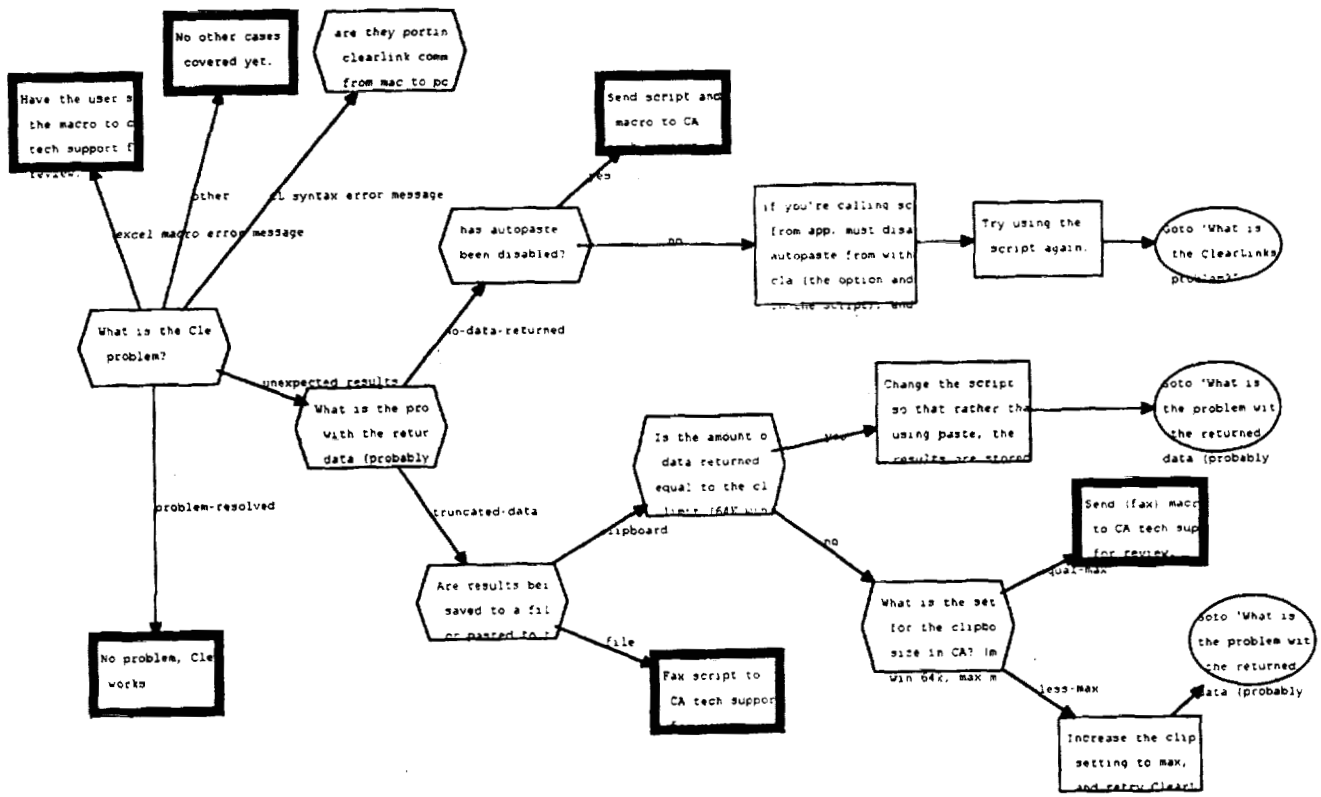

Figure 1. A fragment of the TARGET task network for the ClearAccess technical support expert system.

The network fragment shown in Figure 1 is only a portion of the network for dealing with problems involving ClearAccess and Excel. The total network for dealing with these problems presently has 46 nodes. The fragment pictured has 21 nodes. There are four other problem areas that the expert system currently covers. These areas have their own task networks, each containing from 40 to 50 nodes.

KNOWLEDGE ENGINEERING

The knowledge engineering process involves taking the information gained from the knowledge acquisition process and implementing it in a computer executable form. In this case we are implementing the information as forward-chaining production rules for the CLIPS expert system shell. Typically, the knowledge acquisition process produces textual notes which may or may not be accompanied by graphs, charts, or pseudo-code rules. In the present case, the knowledge acquisition process produces the TARGET task network. We found that translating the task networks into CLIPS rules was straightforward once we established guidelines for creating the TARGET representation.

Rule Creation Phase

Figure 2 shows two rules developed from the partial network shown in Figure 1. The first rule, CLask-clipboard-limit, comes from a hexagonal decision node. The second rule, CLsave-results-to-file comes from the rectangular required-task node that immediatly follows the decision node of the first rule. As illustrated in the first rule, decision nodes become queries when translated to CLIPS rules. The links leaving decision nodes become a menu of possible responses to the query. In this rule, the menu presented to the end user will simply have alternatives of "yes" or "no". This yes-no menu is produced by the *yes* function that appears in the assertion of the rule. The rest of the consequent has the effect of creating an attribute, data-returned-equals-clipboard-limit, and then assigning the appropriate "yes" or "no" value (selected by the end user) to the attribute. This attribute-value pair is then asserted to the fact list. The antecedents of the rule are derived from all the assertions along the path leading to the decision node.

```
(defrule CLask-clipboard-limit
  (phase clinks)
  (subproblem unexpected-results)
  (unexpected-result-type truncated-data)
  (saving-results-file-or-clipbd clipboard)
  =>
  (assert (data-returned-equals-clipboard-limit
    =(yes "Is the amount of data being
          returned equal to the clipboard limit
          (64K: Windows, 32K: Mac)?"))))
```

```
(defrule CLsave-results-to-file
  (phase clinks)
  (subproblem unexpected-results)
  (unexpected-result-type truncated data)
  (data-returned-equals-clipboard-limit yes)
  (saving-results-file-or-clipbd clipboard)
  =>
  (msg "Change the script so that rather than
        using paste, the results are stored to a
        file; then open the file in the application.")
  (assert (saving-results-file-or-clipbd file)))
```

Figure 2. Two CLIPS rules derived from task network displayed in Figure 1.

It is important that all the nodes along the path to the decision node are involved in the antecedent of the rule. This serves to make the rule a self-contained piece of knowledge. This self-containment property is fundamental to having a comprehensible and well-structured rule-based expert system (Brownston, et. al., 1985; Ignizio, 1991; Pedersen, 1989).

The second rule in Figure 2 illustrates that required-task nodes will typically produce some notification message telling the user to perform some action. (Or, in a more automated environment the expert system might perform the action itself). The consequent of the rule must assert a fact onto the fact list to represent that the indicated action has been performed. In this rule a message is given to the user to change the spreadsheet macro-script such that the results are stored to a file rather than using the paste function. A corresponding fact is then asserted to

the fact list. Analogous to the case for decision nodes, the antecedents for this rule are derived from all the assertions along the path leading to the required-task node.

Rule Refinement Phase

A greater challenge than creating a set of rules from the initial TARGET network was keeping the TARGET representation consistent with the CLIPS rules as we refined the system. Such refinements are a major part of the expert system development process. The usual case in developing an expert system is to quickly develop an initial prototype system consisting of a first-pass set of rules. Most of the work in developing the system is then performed as a process of trying out the prototype, adding new rules, and modifying rules as necessary.

The difficulty in maintaining consistency arises because the expert system's behavior results from the knowledge as implemented in CLIPS rules and executed by the CLIPS inference engine. The knowledge underlying the system, however, is most easily analyzed in terms of the graphical TARGET representation. We found that it was quite convenient and informative to use the TARGET network as a guide to which parts of the expert system were incomplete or needed further refinements.

For this knowledge engineering phase we used two PCs operating side-by-side. One was displaying the TARGET network and the other was executing the CLIPS expert system. We used the TARGET network as a navigation guide for testing the behavior of the expert system. We made only minor changes to the CLIPS rules during these sessions. We found it was much more efficient to make changes to the TARGET network during the session with the domain expert rather than to the CLIPS rules. We added new nodes to the network in some cases and in others modified existing nodes. We clearly marked any changes to the network so that we could return to them at a later time and make the necessary modifications to old rules or additions of new rules to the CLIPS rule set.

This procedure was heavily dependent on maintaining a close correspondence between the network and rule representations. If there was not a close correspondence then we could not easily locate the network nodes to modify in response to desired changes in the behavior of the expert system. Or, we might make changes to network nodes, but if there is a lack of correspondence then it becomes quite difficult to locate the CLIPS rules that need to be changed in response to the network changes. We found that unless we were very careful and conscientious about maintaining this consistency the two representations quickly diverged and it became impossible to relate them.

VALIDATION AND DOCUMENTATION

Maintaining consistency between the network and rule representations was a lot of work. It required a significant amount of discipline and effort on our part. This was not surprising since the task network represents a type of external documentation for the expert system code. Most software developers tend to avoid careful documentation of their code unless they are required to do so. Such writing takes time and effort, and developers almost always feel that their code is clear and comprehensible--at least at the time they are writing it. This natural tendency to avoid documentation is even stronger in developing expert systems where one uses a fast prototyping approach and code is typically written, tested, and then revised or discarded. It is almost always the case, however, that the code is never so clear and comprehensible to other individuals who might have to maintain the code at a later date. Further, it is common experience that the code becomes difficult to understand even for the original developer after a few weeks or ever a few days away from it.

Thus, one significant benefit of maintaining consistency of the network and rule representations beyond its immediate benefit of aiding the rule refinement process is that it results in effective documentation of the knowledge in the expert system. Indeed, the two representations document the knowledge in the system in complementary ways. Rule-based representations are well-suited to documenting self-contained chunks of declarative knowledge (Brownston et. al., 1985; Ignizio, 1991; Pedersen, 1989). But they are not so appropriate for documenting the procedural flow-of-control among the chunks of knowledge. This information is implicit in the interaction between the entire rule set and the expert system inference engine. The task network representation, on the other hand, is well-suited to documenting the flow of control of the knowledge in the system. It is not so strong at representing the self-contained chunks of knowledge that are well-captured by production rules.

This documentation of corporate knowledge is by itself of significant value to a company. It is also quite useful for validating the knowledge in the implemented version of the system. Given a task network representation that is closely related to the rule-based implementation of the same knowledge, it is quite straightforward to use the network to guide a domain expert in testing the system such that he or she evaluates at least one path to every final state in the system. Referring to the network fragment in Figure 1, one can see that it would be straightforward to use the network to direct answers to the expert system queries such that the network is navigated to each terminal node. After a terminal node is encountered the knowledge engineer (or domain expert) marks that node and its incoming link as having already been traversed. In this way it is easy to ensure that every terminal node has been visited at least once.

Unless the task network has a simple tree topology, this procedure does not guarantee that all possible paths through the expert system have been examined. This would require a more sophisticated graph traversal procedure. However, even this simple procedure should guarantee a much more systematic and thorough validation check of an expert system than is often done for rule-based expert systems.

SUMMARY AND CONCLUSION

First, we found TARGET to be very helpful in the knowledge acquisition process. It enabled us to perform knowledge acquisition with one knowledge engineer rather than two, and in addition it improved communication between the domain expert and knowledge engineer. We also found it to be useful for both the rule development and refinement phases of the knowledge engineering process. Its usefulness in this process required that we adopt guidelines for developing the network. These guidelines enabled us to easily translate the network into production rules. A significant requirement for TARGET remaining useful throughout the knowledge engineering process was the need to carefully maintain consistency between the network and the rule representations. Maintaining such a consistency not only benefited the knowledge engineering process, but also had significant payoffs in the areas of validation of the expert system and documentation of the knowledge in the system. We conclude that TARGET is a very useful tool for aiding the development of rule-based expert systems such as are typically developed with the CLIPS expert system shell.

Finally, we propose that a promising area for future work is to automate the process of generating CLIPS rules from a TARGET task network. Such a facility would simultaneously eliminate the need to hand-code CLIPS rules and the problem of maintaining consistency between the task network and the production rules. The generality of such a rule generator will depend on the specific guidelines and restrictions that are placed on the construction of the task networks.

ACKNOWLEDGMENTS

Ahto Jarve provided very able assistance in performing much of the knowledge acquisition and knowledge engineering for the expert system described in the paper. Melinda Thomas and Patrick Geurin of Fairfield Software provided the domain expertise encoded in the expert system. I would also like to acknowledge the initial work on the Clear Access technical support expert system carried out by Dr. Ralph Bunker, Patrick Daley, and Slobodan Dumuzliski.

REFERENCES

Arnold, B., "Expert System Tools Optimizing Help Desks," SOFTWARE MAGAZINE, January, 1993, pp. 56 - 64.

Brownston, L., Farrell, R., Kant, E., and Martin, N., PROGRAMMING EXPERT SYSTEMS IN OPS5, Addison-Wesley, Reading, MA, 1985.

ClearAccess, "User's Guide," ClearAccess Corporation, Fairfield, Iowa, 1994.

Hayes-Roth, F. Waterman, D.A., and Lenat, D., BUILDING EXPERT SYSTEMS, Addison-Wesley, Reading, MA, 1983.

Hedberg, S. "The Evolution of Applied AI: A Report on the Fifth Innovative Applications of AI Conference," AI MAGAZINE, Fall, 1993.

Ignizio, J., INTRODUCTION TO EXPERT SYSTEMS: THE DEVELOPMENT AND IMPLEMENTATION OF RULE-BASED EXPERT SYSTEMS, McGraw-Hill, 1991.

Ortiz, C.J., Ly, H.V., Saito, T., and Loftin, R.B. "TARGET: Rapid Capture of Process Knowledge," PROCEEDINGS OF TECHNOLOGY 2002: THE THIRD NATIONAL TECHNOLOGY TRANSFER CONFERENCE AND EXPOSITION, Vol 1, NASA, Washington, February 1993, pp 279-288.

Pedersen, K., EXPERT SYSTEMS PROGRAMMING: PRACTICAL TECHNIQUES FOR RULE-BASED SYSTEMS, Wiley, 1989.

NASA, "TARGET Reference Manual," Software Technology Branch, NASA, Johnson Space Center, Houston, Texas, November, 1993.

Rewari, A., "AI in Corporate Service and Support," IEEE EXPERT, Vol 8:6, 1993, pp 5 - 40.