N90-27307

THE INTEGRATION OF AUTOMATED KNOWLEDGE ACQUISITION WITH COMPUTER-AIDED SOFTWARE ENGINEERING FOR SPACE SHUTTLE EXPERT SYSTEMS

Dr. Kenneth L. Modesitt
Head, Department of Computer Science
Western Kentucky University
Bowling Green, KY 42101

ABSTRACT: The phrase "expert systems" will disappear within ten years. Somewhat less likely to suffer the same fate will be the term "knowledge acquisition." The field of software engineering will expand to include both of these terms, wherein expert systems will be a form of advanced software engineering. The incorporation will permit more complex domains to be addressed, and the unique qualities of expert systems will render the resulting software more transparent. System specification and requirements analysis will be augmented by knowledge acquisition techniques to enable prototypes to appear earlier for customer inspection, with the end result being a software product for which the customer has a real need, and which performs up to her expectations (Ref. 2)

The current qualities of expert systems will become embodied in various components of software engineering methodologies and end products. The most likely candidate for this process in Computer Aided Software Engineering (CASE) tools. For once, we in the software engineering world will not have to continue to be the shoemaker's children. We have constructed powerful, useful, and extensible automated tools for our own use, rather than only building them for others. The features of expert systems will be used in most parts of CASE, including needs assessment, requirements analysis, design, implementation, testing, and maintenance and enhancement. Project planning, documentation and software quality assurance will also benefit. The growing interest in "reverse" software engineering, of going from existing ill-structured and non-documented code to modular design representations, will be a ripe field for expert system contributions. The critical nature of user interfaces will be addressed by our expertise in transparency and explanation-based learning of expert systems.

Many of the above "predictions" are not really futuristic at all. They were incarnated in the process of constructing an automated test analysis computer system for the Space Shuttle Main Engine (SSME) by the Rocketdyne Division of Rockwell International (Refs. 3,4,5). The development effort was successful in bringing the system SCOTTY on-line in June, 1988 at somewhat over 25% of the eventual full system, in terms of thoroughness of the SSME test analysis procedure. Progress has continued to date, and has spawned other automated SSME systems, plus ones related to other Rocketdyne programs such as expendable launch vehichles, the engines for the National Aerospace Plane, and the Space Station power system.

This successful development was made possible by an optimal mix of vision, personnel, tools, procedures and management. The personnel included an excellent young mechanical and aerospace engineering staff, and a knowledge engineer with both industry and academic credentials. The tools were recommended by the knowledge engineer, and included both an industrial-strength inductive Expert System Building Tool, running on a multi-processor supermini computer from Concurrent Computer Corporation, as well as a PC-based CASE tool.

Management direction was given by an enthusiastic senior technical manager who was very well respected in the company, and who had realistic expectations about expert system abilities. He also ensured that the personnel and financial committments to the program were long-term ones.

Since the knowledge engineer had a substantial background in software engineering, both as a practicing professional and as an academic since 1963, it was natural that the "front-end" of the development effort would receive considerable attention. The desireability of this front-loading has manifested itself innumerable times throughout the software industry in the savings accrued in the "back-end" of the software life cycle. Maintenance, including all three types: corrective, perfective, and adaptive, has long been recognized as the real cost driver in software.

Consequently, a great deal of attention was paid to the interactions among the expert, the user, the protoype system, and the knowledge engineer. Many alternatives were considered for this knowledge acquisition process. The recent book by Karen McGraw and Karan Harbison-Briggs (Ref. 1), with a preface by this author, would have been invaluable. Figure 1 for some knowledge acquisition alternatives is from the book.

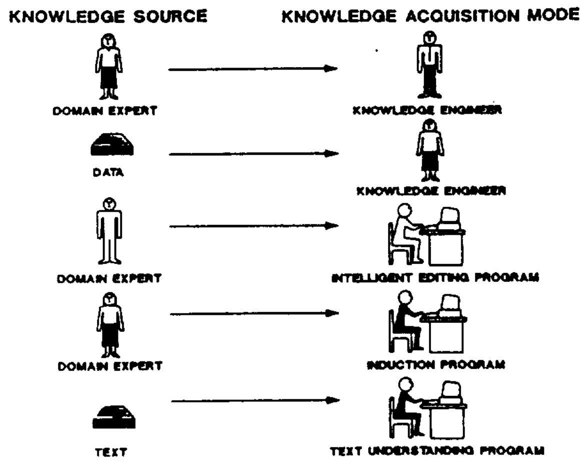

Figure 1. Variations in Possible Knowledge Acquisition Mode
(Reprinted with permission of Prentice-Hall from Knowledge Acquisition: Principles and Guidelines by Karen McGraw and Karan Harbison-Briggs, 1989.)

Initially, in 1984, a small PC-based inductive system was used to demonstrate a feasibility prototype. This took only a few days, with the expert quickly learning the tool, and appreciating the power, vs. having to codify his own rules. The order-of-magnitude increase of having the expert express his knowledge as examples, and then having the ESBT generating the rules was obvious, and has been well-documented many times since then. It was also obvious that a more powerful tool would be required for a production test analysis system. The vendor of the small tool was just ready to announce such a product, which was a near-ideal fit. In addition to induction, it also generated Fortran code, which is the lingua franca of the engineering world. This is critical as the code output by the ESBT is readable, i.e., not "magic", and it is trivial to interface it to the 100K+ lines of Fortran code which already exist for SSME software support, plus new codes which would surely be written in the future.

Good management practices and documentation guidelines dictated that all of this effort be tracked. Requirement documents were generated, as were data flow diagrams and structure charts. These were invaluable in not becoming lost as SCOTTY grew in complexity. Moreover, the tools used in CASE were easily grasped by the mechanical engineers. The fact that expert system tools were being used did not obviate the fact that it was still very much of a software engineering process, albeit in a complex domain.

In the future, it is clear that expert systems and software engineering will intertwine even more closely. A few such considerations include data bases, and automatic code generation from structure chart modules. In the case of the latter, the author and a colleague were the first to build an interface which permitted the ESBT to interact with the millions of bytes of test data from the 1000+ previous SSME tests. This interface has now been expanded by a joint venture between Intelligent Terminals Ltd. and Concurrent Computer Corporation to become a commercial product. In the case of the latter feature -- automatic code generation -- it will not be long before a CASE vendor adds inductive programming to the tool chest.

In summary, a prediction was made that the terms "expert systems" and "knowledge acquisition" would begin to disappear over the next several years. This is not because they are falling into disuse; it is rather that practitioners are realizing that they are valuable adjuncts to software engineering, in terms of problem domains addressed, user acceptance, and in development methodologies. A specific problem domain was discussed, that of constructing an automated test analysis system for the Space Shuttle Main Engine. In this domain, knowledge acquisition was part of requirements systems analysis, and was performed with the aid of a powerful inductive ESBT in conjunction with a CASE tool. The original prediction is not a very risky one -- it has already been accomplished!

References

1. McGraw, K. and K. Harbison-Briggs, Knowledge Acquisition: Principles and Guidelines, Prentice-Hall, 1989.

2. Modesitt, K., "Inductive Learning in Engineering". Tutorial for the International Special Interest Group on Inductive Programming, Detroit, MI, April, 1989.

3. Modesitt, K., "Experience with Commercial Tools Involving Induction on Large Databases for Space Shuttle Main Engine Testing," Invited talk for Fourth International Expert Systems Conference, London, England, 1988, pp. 219-229.

4. Modesitt, K., "Space Shuttle Main Engine Anomaly Data and Inductive Knowledge Based Systems: Automated Corporate Expertise," Third Conference on Artificial Intelligence for Space Applications, NASA, Huntsville, 1987, pp. 203-212.

5. Modesitt, K., "Space Shuttle Main Engine Test Analysis -- A Case Study for Inductive Knowledge-based Systems Involving Very Large Data Bases," Co-authored with Dr. Djamshid Asgari. IEEE Computer Society International Conference on Computers and Application Conference (COMPSAC), Chicago, October, 1986, pp. 65-71.