N88-14875 S1-63

116664
158

A GENERAL ARCHITECTURE FOR INTELLIGENT TRAINING SYSTEMS

Final Report

NASA/ASEE Summer Faculty Fellowship Program--1987

Johnson Space Center

Prepared by:	R. Bowen Loftin, Ph.D.
Academic Rank:	Associate Professor
University & Department	University of Houston-Downtown Department of Natural Sciences One Main Street Houston, Texas 77002
NASA/JSC	
Directorate:	Mission Support
Division:	Mission Planning and Analysis
Branch:	Technology Development and Applications
Section:	Artificial Intelligence
JSC Colleague:	Robert T. Savely
Date:	August 14, 1987
Contract Number:	NGT 44-001-800

ABSTRACT

A preliminary design of a general architecture for autonomous intelligent training systems has been developed. The architecture integrates expert system technology with teaching/training methodologies to permit the production of systems suitable for use by NASA, other government agencies, industry, and academia in the training of personnel for the performance of complex, mission-critical tasks. The proposed architecture consists of five elements: a user interface, a domain expert, a training session manager, a trainee model, and a training scenario generator. A user interface permits the trainee to access data and perform actions as he would in the task environment. The interface also provides the trainee with information on the current training environment and with on-line help. The domain expert contains the rules and procedural knowledge needed to carry out the task. It also includes "mal-rules" which permit the identification of common errors on the part of the trainee. The training session manager (TSM) examines the actions of the trainee and compares them with the actions of the domain expert. A unique feature of the TSM is its ability to permit the trainee the freedom to follow any valid path between two stages of the task for which he is being trained. Following each trainee action, evaluative assertions are made by the TSM and used to update the trainee model. A trainee model is developed for each individual using the system. The model includes a history of the trainee's interactions with the intelligent training system and provides evaluative data on the trainee's current skill level. A training scenario generator designs appropriate training exercises for each trainee based on the trainee model and the training goal(s). The design of this architecture has been guided and its efficacy tested through the development of a system for use by Mission Control Center Flight Dynamics Officers in training to perform Payload-Assist Module Deploys from the orbiter.

INTRODUCTION

Government and industry must maintain a large effort in the training of their personnel: new personnel must be trained to perform the task(s) which they were hired to perform, continuing personnel must be trained to upgrade or update their ability to perform assigned tasks, and continuing personnel must be trained to tackle new tasks. Within NASA a great number of training methodologies are employed, singly or in concert. These methods include training manuals, formal classes, procedural computer programs, simulations, and on-the-job training. The latter method is particularly effective in complex tasks where a great deal of independence is granted to the task performer. Of course, this training method is also the most expensive and may be impractical when there are many trainees and few experienced personnel to conduct on-the-job training.

This report describes an effort to design an architecture for an entire generation of autonomous training systems which integrate many of the features of "traditional" training programs with expert system technology. The ultimate goal of this program is a software development environment which would permit those responsible for training in government, industry, and academia to develop intelligent computer-aided training (ICAT) systems for specific tasks. These ICAT systems would operate autonomously and would provide, for the trainee, much of the same experience that could be gained from the best on-the-job training. By integrating domain expertise with a knowledge of appropriate training methods, an ICAT session is intended to duplicate, as closely as possible, the trainee undergoing on-the-job training in the task environment, benefiting from the full attention of a task expert who is also an expert trainer. Thus, the philosophy of the ICAT system is to emulate, to the extent possible, the behavior of an experienced individual devoting his full time and attention to the training of a novice--proposing challenging training scenarios, monitoring and evaluating the actions of the trainee, providing meaningful comments in response to trainee errors, responding to trainee requests for information and hints (if appropriate), and remembering the strengths and weaknesses displayed by the trainee so that appropriate future exercises can be designed.

BACKGROUND

Since the 1970's a number of academic and industrial researchers have explored the application of artificial intelligence concepts to the task of teaching a variety of subjects (e.g., geometry, computer programming, medical diagnosis, and electronic troubleshooting). A body of literature is now extant on student models and teaching/tutoring methodologies adapted to intelligent tutoring systems in the academic environment[1]. The earliest published reports which suggested the applications of artificial intelligence concepts to teaching tasks appeared in the early 1970's.[2,3] Hartley and Sleeman[3] actually proposed an architecture for an intelligent tutoring system. However, it is interesting to note that, in the fourteen years which have passed since the appearance of the Hartley and Sleeman proposal, no agreement has been reached among researchers on a general architecture for intelligent tutoring systems.[4] Nonetheless, a study of the literature on intelligent tutoring systems is an essential starting point for the development of the elements of an intelligent training system.

Among the more notable intelligent tutoring systems reported to date are SOPHIE[5], PROUST[6] and the LISP Tutor[7]. The first of these systems, SOPHIE, was developed in response to a U.S. Air Force interest in a computer-based training course in electronic troubleshooting. SOPHIE contains three major components: an electronics expert with a general knowledge of electronic circuits, together with detailed knowledge about a particular type of circuit (in SOPHIE this was an IP-28 regulated power supply); a coach which examines student inputs and decides if it is appropriate to stop the student and offer advice; and a troubleshooting expert that uses the electronics expert to determine which possible measurements are most useful in a particular context. Three versions of SOPHIE were produced and used for a time but none was ever viewed as a "finished" product. One of the major lacks of the SOPHIE systems was a user model. It is interesting to note that the development of a natural language interface for SOPHIE represented a large portion of the total task.

PROUST and the LISP Tutor are two well-known intelligent tutoring systems that have left the laboratory and found wider applications. PROUST (and its offspring, Micro-PROUST) serves as a "debugger" for finding nonsyntactical errors in Pascal programs written by student programmers. The developers of PROUST claim that it is capable of finding all of the bugs in at least seventy percent of the "moderately complex" programming assignments that its examines. PROUST contains an expert Pascal programmer that can write "good" programs for the assignments given to students. Bugs are found by matching the expert's program with that of the student; mismatches are identified as "bugs" in the student program. This ability is contained in the PROUST "bug rule" component. After finding a bug, PROUST provides an English-language description of the bug to the student, enabling the student to correct his error. The system cannot handle student programs that depart radically from the programming "style" of the expert. The LISP Tutor is currently used to teach the introductory Lisp course offered at Carnegie-Mellon University. This system is based on the ACT (historically, Adaptive Control of Thought) theory and consists of four elements: a structured editor which serves as an interface to the system for students, an expert Lisp programmer that provides an "ideal" solution to a programming problem, a bug catalog that contains errors made by novice programmers, and a tutoring component that provides both immediate feedback and guidance to the student. Evaluations of the LISP Tutor show that it can achieve results similar to those obtained by human tutors. One of its primary features is its enforcement of what its authors regard as a "good" programming style.

TRAINING VERSUS TUTORING

The ICAT architecture has been developed with a clear understanding that training is not the same as teaching or tutoring.[8] The NASA training environment differs in many ways from an academic teaching environment These differences are important in the design of an architecture for an intelligent training system:

a. Assigned tasks are often mission-critical, placing the responsibility for lives and property in the hands of those who have been trained.

b. Personnel already have significant academic and practical experience to bring to bear on their assigned

c. Trainees make use of a wide variety of training techniques, ranging from the study of comprehensive training manuals to simulations to actual on-the-job training under the supervision of more experienced

d. Many of the tasks offer considerable freedom in the exact manner in which they may be accomplished.

Those undergoing training for complex, mission-critical tasks are usually well aware of the importance of their job and the probable consequences of failure. While students are often motivated by the fear of receiving a low grade, trainees know that human lives and/or expensive equipment may depend on their skill in performing assigned tasks. This means that trainees may be highly motivated, but it also imposes on the trainer the responsibility for the accuracy of the training content (i.e., verification of the domain expertise encoded in the system) and the ability of the trainer to correctly evaluate trainee actions. The ICAT system is intended, not to impart basic knowledge, but to aid the trainee in developing skills for which he already has the basic or "theoretical" knowledge. In short, this training system is designed to help a trainee put into practice that which he already intellectually understands. The system must take into account the type of training that both precedes and follows, building on the knowledge gained from training manuals and rule books while preparing the trainee for and complementing the on-the-job training which will follow. Perhaps most critical of all, trainees must be allowed to carry out an assigned task by any valid means. Such flexibility is essential so that trainees are able to retain and even hone an independence of thought and develop confidence in their ability to respond to problems, even problems which they have never encountered and which their trainers never anticipated.

SYSTEM DESIGN

The ICAT system architecture is modular and consists of five basic components:

1. A user interface that permits the trainee to access the same information available to him in the the task environment and serves as a means for the trainee to assert actions and communicate with the intelligent training system
2. A domain expert which can carry out the task using the same information that is available to the trainee and which also contains a list of "mal-rules" (explicitly identified errors that novice trainees commonly make).
3. A training session manager (TSM) which examines the assertions made by the domain expert (of both correct and incorrect actions in a particular context) and by the trainee. Evaluative assertions are made following each trainee action. In addition, guidance can be provided to the trainee if appropriate for his skill level.
4. A trainee model which contains a history of the individual trainee's interactions with the system together with summary evaluative data.
5. A training scenario generator that designs increasingly-complex training exercises based on the current skill level contained in the trainee's model and on any weaknesses or deficiencies that the trainee has exhibited in previous interactions.

Figure 1 contains a schematic diagram of the ICAT system. Note that provision is made for the user to interact with the system in two distinct ways and that a supervisor may also query the system for evaluative data on each trainee. The blackboard serves as a common "factbase" for all five system components. With the exception of the trainee model, each component makes assertions to the blackboard, and the rule-based components look to the blackboard for facts against which to pattern match the left-hand sides of their rules.

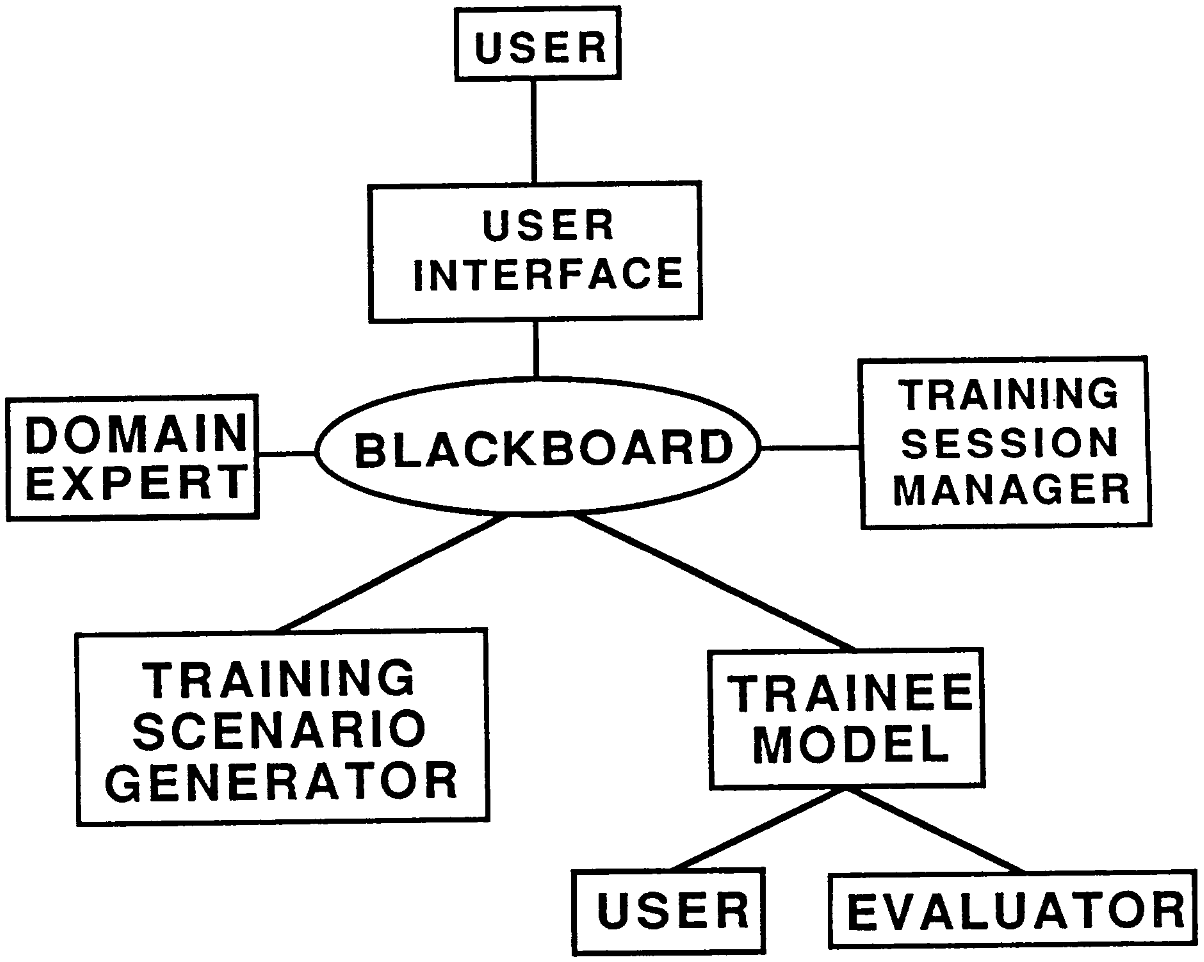

FIGURE 1. ICAT ARCHITECTURE

User Interface

The primary factor influencing the interface design is fidelity to the task environment. To avoid negative training, it is essential that the functionality and, to the extent possible, the actual appearance of the training environment duplicate that in which the task is performed. The interaction of the user interface with the other components of the ICAT system can be standardized and transported to any task environment; however, the user interface, as viewed by the trainee, must be developed for each task environment. The training of different tasks in the same environment may be able to utilize the same user interface. As a part of this project, a user interface was designed for a console position in the Mission Control Center at NASA/Johnson Space Center. The details of that interface are described elsewhere.[9]

Domain Expert

The domain expert is a "traditional" expert system in that it contains if-then rules which access data describing the task environment and is capable of executing the task and arriving at the correct "answers" or performing the correct actions. In addition to "knowing" the right way to carry out the task, the domain expert also contains knowledge of the typical errors that are made by novices. In this way, the ICAT system can not only detect an erroneous action made by a trainee, but also, through these so-called "mal-rules", it can diagnose the nature of the error and provide an error message to the trainee specifically designed to inform the trainee about the exact error made and correct the misconception or lack of knowledge that led to the commission of that error. Another of the interesting features of the ICAT system is its continual awareness of the environment (the external constraints dictated by the training exercise) and the context of the exercise. Rather than having the domain expert generate a complete and correct set of actions to accomplish the task, only those actions which are germane to the current context are asserted. In this way the expert "adapts" to alternate, but correct, paths that the trainee might choose to follow. Figure 2 shows schematically how the domain expert operates. This strategy was adopted because the human experts that perform complex tasks often recognize that many steps in the process may

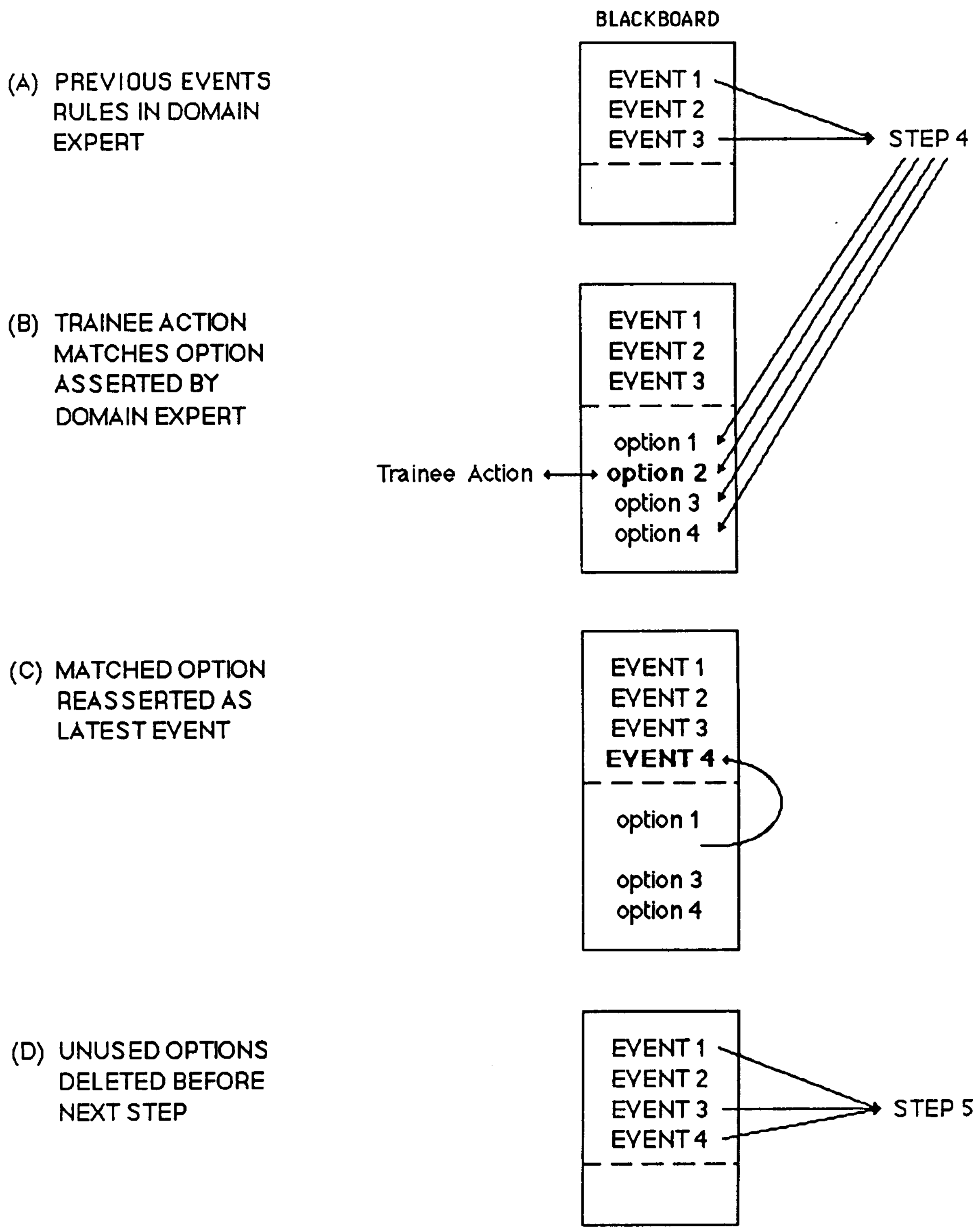

FIGURE 2. OPERATION OF DOMAIN EXPERT

be accomplished by two or more equally valid sequences of actions. To grant freedom of choice to the trainee and to encourage independence on his part, this type of flexibility in the ICAT system was deemed essential.

Training Session Manager

The training session manager is dedicated principally to error-handling. Its rules compare the assertions of the domain expert with those of the trainee to detect errors. Subsequently, the domain expert asserts facts that allow the TSM to write appropriate error messages to the trainee through the user interface. In addition, the TSM is sensitive to the skill level of the trainee as represented by the trainee model. As a result, the detail and "tone" of error messages is chosen to match the current trainee. For example, an error made by a first time user of the training system may require a verbose explanation so that the system can be certain that the trainee will have all of the knowledge and concepts needed to proceed. On the other hand, an experienced trainee may have momentarily forgotten a particular procedure or may have "lost his place". In this latter case a terse error message would be adequate to allow the trainee to resume the exercise. The TSM also encodes all trainee actions, both correct and incorrect, and passes them to the trainee model.

Trainee Model

Successful intelligent tutors incorporate student models to aid in error diagnosis and to guide the student's progress through the tutor's curriculum.[10] The trainee model in the ICAT system stores assertions made by the TSM as a result of trainee actions. Thus, at its most fundamental level, the trainee model contains, for the current training session, a complete record of the correct and incorrect actions taken by the trainee. At the conclusion of each training session, the model updates a training summary which contains information about the trainee's progress, such as a skill level designator, number of sessions completed, number of errors made (by error type and session), and the time taken to complete sessions. After completing a session, the trainee can obtain a report of that session which contains a comprehensive list of correct and incorrect actions together with an evaluative commentary. A supervisor can access each trainee's model to obtain this same report or to obtain summary data, at a higher level, on the trainee's progress. Finally, the training scenario generator uses the trainee model to produce new training exercises.

Training Scenario Generator

The training scenario generator relies upon a database of task "problems" to structure unique exercises for a trainee each time he interacts with the system. The initial exercises provided to a new trainee can be based on variants of the purely nominal task with no time constraints, distractions or "problems". Once the trainee has demonstrated an acceptable level of competence with the nominal task, the generator draws upon its database to insert selected problems into the training environment. In addition, time constraints are "tightened" as the trainee gains more experience and distractions of a form appropriate for the task environment are presented at "inconvenient" points during the task. The generator also examines the trainee model for particular types of errors committed by the trainee in previous (and the current) sessions. The trainee is then given the opportunity to demonstrate that he will not make that error again. Ultimately, the trainee is presented with exercises which embody the most difficult problems together with time constraints and distractions comparable to those encountered during the completion of an experienced person in the actual task environment.

SYSTEM INTEGRATION

The ICAT system which has been built as a prototype for this project is currently operational on a Symbolics 3600 series Lisp machine. The user interface and trainee model are written in common Lisp while the rules of the domain expert, TSM, and the training scenario generator are written in ART 3.0. This prototype system will ultimately be delivered to the users in a Unix workstation environment. To accomplish this delivery, the ART rules were written to facilitate translation into to CLIPS[11] and the Lisp code will be converted into C.

CONCLUSIONS

The prototype ICAT system has, so far, proven to be a potentially valuable addition to the training tools available for training Flight Dynamics Officers in shuttle ground control. The authors are convinced that the basic structure of the ICAT system described here can be extended to form a general architecture for intelligent training systems for training flight controllers and crew members in the performance of complex, mission-critical tasks. It may ultimately be effective in training personnel for a wide variety of tasks in governmental, academic, and industrial settings.

REFERENCES

1. See, for example, Sleeman, D. and Brown, J.S. (eds.), Intelligent Tutoring Systems (London: Academic Press, 1982) and Yazdani, M. "Intelligent Tutoring Systems Survey," Artificial Intelligence Review **1**, 43 (1986).

2. Carbonell, J.R. "AI in CAI: An Artificial Intelligence Approach to CAI," IEEE Transactions on Man-Machine Systems **11**(4), 190 (1970).

3. Hartley, J.R. and Sleeman, D.H., "Towards Intelligent Teaching Systems," International Journal of Man-Machine Studies **5**, 215 (1973).

4. Yazdani, M. "Intelligent Tutoring Systems Survey," Artificial Intelligence Review **1**, 43 (1986).

5. Brown, J.S., Burton, R.R., and de Kleer, J., "Pedagogical, Natural Language and Knowledge Engineering Techniques in SOPHIE I, II, and III," in Sleeman, D. and Brown, J.S., (eds.), Intelligent Tutoring Systems (London: Academic Press, 1982), p. 227.

6. Johnson, W.L. and Soloway, E. "PROUST, " Byte **10** (4), 179 (April, 1985).

7. Anderson, J.R., Boyle, C.F., and Reiser, B.J., "Intelligent Tutoring Systems," Science **228**, 456 (1985) and Anderson, J.R. and Reiser, B.J., "The LISP Tutor," Byte **10**(4), 159 (April, 1985).

8. Harmon, P. "Intelligent Job Aids: How AI Will Change Training in the Next Five Years," in Kearsley, G., ed., Artificial Intelligence and Instruction: Applications and Methods (Reading, MA: Addison Wesley Publishing Co., 1987).

9. Loftin, R.B., Wang, L., Baffes, P., and Rua, M., "An Intelligent Computer-Aided Training System for Payload-Assist Module Deploys," Proceedings of the First Annual Workshop on Space Operations, Automation & Robotics (SOAR '87), NASA/Johnson Space Center, Houston, TX, August 4-6, 1987.

10. See, for example, a number of papers on student models in Sleeman, D. and Brown, J.S., (eds.), Intelligent Tutoring Systems (London: Academic Press, 1982)

11. "CLIPS" is an acronym for "C-Language Integrated Production System" and was developed by the Artificial Intelligence Section, Mail Code FM72, NASA/Johnson Space Center, Houston, TX 77058. Its advantages as a delivery vehicle for expert systems are discussed in Giarratano, J., Culbert, C., Riley, G., and Savely, R.T., "A Solution of the Expert System Delivery Problem," submitted for publication in IEEE Expert. For additional information on CLIPS, write to the AI Section at NASA/JSC.