Rapid Development of Scenario-Based Simulations and Tutoring Systems

John L. Mohammed, Ph.D.[*], James C. Ong[†], and Jian Li[‡]
Stottler Henke Associates Inc., San Mateo, CA, 94404

and

H. Barbara Sorensen, Ph.D.[§]
Air Force Research Laboratory, Mesa, AZ, 85212

Low earth orbit constellations significantly increase the burden for ground-based systems management. Efficient, accurate management requires that ground station and satellite operators are well-trained. Scenario-based training, in which trainees practice handling specific situations using faithful simulations of the equipment they will use on the job has proven to be an extremely effective method for both training and certification. Effectiveness is increased when it is coupled with intelligent tutoring. Intelligent tutoring systems (ITS) improve on training simulations by assessing the trainee's actions and providing individualized guidance. Development of training simulators often lags behind development of the operational systems, with the result that simulators are often not available or up to date when they are most needed (e.g., when training operators on new or modified systems). Also, practical deployment of intelligent tutoring systems is hindered by the time and expense required to prepare the needed knowledge representations. We report on the development of tools to address both these issues. Three strategies enable rapid development. For both simulation and tutoring, the first strategy is to develop systems that are scenario-specific. This strategy permits the systems to be simpler and easier to develop. The second strategy is to employ easily manipulated representations for the simulation behavior and tutoring knowledge, and provide graphical user interfaces to edit these representations that facilitate development by training instructors with little outside support. The third strategy is to minimize the need for development. For simulation, this means maintaining a high degree of fidelity with little programming by incorporating screen captures of the operational system, and by enabling integration of existing simulation codes. For tutoring, this means capturing the bulk of the tutoring knowledge by simply recording the actions of instructors as they demonstrate the correct manner of handling the scenario. These strategies are embodied in two systems developed by Stottler Henke for the Air Force. TaskSim is a simulation framework to facilitate rapid development of scenario-based training simulations. Task Tutor Toolkit™ is an intelligent tutoring system that enables instructors to create scenario-specific tutors through a three-step process: demonstrate, generalize and annotate.

I. Introduction

Increasingly, military and commercial satellite systems are employing constellations of satellites in low earth orbit (LEO) for communications and remote sensing. Satellite system management is complicated by the large number of satellites to be managed and the brief time windows when each satellite is visible to ground communication sites during which communication can take place. Therefore, it is essential that operators make the best use of every opportunity to communicate with each satellite as it comes into view. Extensive simulation-based training with

[*] Project Manager, 7533 Shelby Street, Elk Grove, CA, 95758
[†] Group Manager, 951 Mariner's Island Blvd. Suite 360, San Mateo, CA 94404
[‡] Software Engineer, 951 Mariner's Island Blvd. Suite 360, San Mateo, CA 94404
[§] Senior Research Scientist, Warfighter Readiness Research Division, 6030 South Kent Street, Mesa, AZ 85212

instructional feedback can prepare trainee operators with repeated practice and exposure to a wide range of nominal and off-nominal situations.

This paper describes a simulation-based intelligent tutoring system developed by the authors to support satellite operations and other complex procedural tasks. These systems rely on computable task representations that specify appropriate actions at each step. These task representations must be expressive enough to enable detailed, context-sensitive guidance and feedback, handle the wide range of situations and anomalies that might occur, and accurately assess the many possible actions the trainee might take. Yet, they must also enable easy and rapid knowledge entry and maintenance of large collections training scenarios.

II. Scenario-Based Intelligent Tutoring

The advantages of scenario-based training are well known.[1] The trainee practices performing tasks in a realistic simulation of the operational environment, receives exposure to a variety of nominal and unusual situations, and gets an opportunity to see how classroom knowledge is applied in context. Simulated scenarios are also a critical part of evaluating trainee performance for certification.

Intelligent tutoring systems (ITS's) can significantly improve the effectiveness of scenario-based training by providing instructional feedback that helps trainees learn from their experiences more reliably. ITS's can track the trainee's progress during the execution of a training scenario. They can be configured to give in situ coaching during exercises such as hints and detailed instructions for what to do, how to do it, and why. ITS's can also assess the trainee's actions, identify areas of strong and weak performance and provide feedback after the trainee completes the scenario. ITS's enable each trainee to receive individualized training that would normally require the full attention of a human tutor — without requiring one instructor per trainee. ITS's also enable the trainee's training to proceed at a pace that is suitable for that particular trainee. By reducing the need for specialized equipment and team members during training exercises, it can also provide increased flexibility regarding when and where training takes place.

Intelligent tutoring systems (ITS's) encode and apply the subject matter teaching expertise of experienced instructors to provide trainees with individualized instruction automatically. For procedural skills such as executing satellite command plans, this expertise includes task knowledge that enables the ITS to evaluate the appropriateness of the trainees' actions and assess their knowledge and skills.

To support training for satellite operations and other procedural tasks, we enhanced a tutoring system and authoring tool called the Task Tutor Toolkit that was originally developed for NASA to support remote payload operations and other technical training areas.[2] This system encodes task knowledge as scenario-specific solution templates that encode allowable sequences of actions for each scenario.

During each exercise, the simulator uses the tutoring system's application programming interface (API) to notify the tutoring system of each trainee action. The simulator also provides query access to simulation state variable values that the tutor can consider when determining the appropriateness of each trainee action. Each action is encoded as a tuple that specifies the type of action and zero or more parameters. For example, setting the oven temperature to 300 degrees might be represented as:

set-control("temperature", 300)

In this example, set-control is the type of action. Two parameters, "temperature" and 300 specify the type of control and the setting, respectively.

A. Hinting

At each step, the trainee can request hints by pressing buttons in the tutoring system window:

- **Give me a hint** – The tutoring system provides an indirect hint that helps the trainee determine an appropriate next action to take.
- **What do I do?** – The tutoring system recommends an appropriate action.
- **How do I do that?** – The tutoring system describes how the trainee should carry out the recommended action using the simulator.
- **Why do I do that?** – The tutoring system explains why the recommended action should be taken. This explanation may be scenario-specific, or it may describe general principles associated with the recommended action.

B. Evaluating Trainee Actions

The tutoring system evaluates each action by comparing it with the scenario's solution template. After each action taken by the trainee, the system displays whether the trainee's action was:

- **Expected** - the action matches an action pattern in the solution template, and the trainee has already carried out all prerequisite actions that should precede this action. For example, an action pattern might match the setting of the temperature control to any value between 290 and 310 degrees.
- **Unexpected** - the action does not match any action pattern in the solution template, or the action has already been carried out, or not all prerequisite steps have been carried out. When a trainee carries out an unexpected action, that action may change the state of the simulated world in a way that invalidates the template's expectations for appropriate next steps. In these situations, the solution template may become invalid, and the tutoring system may no longer be able to assess subsequent trainee actions.
- **Continuable** – the action is unexpected but benign, so the action did not change the state of the simulated world in a way that invalidated the solution template's expectations. The trainee can proceed with the scenario, and the tutor can continue to rely on the solution template to correctly evaluate subsequent actions.
- **Incorrect** - the action and current simulation state match an action pattern and simulation condition, if any, specified within an error rule.

C. Instructional Strategies for Procedural Training

By classifying each trainee action into one of these categories, the tutoring system can support several different instructional strategies. For example, a tutor could accept only expected and continuable actions and reject unexpected and incorrect actions by notifying the trainee and then instructing the simulator to undo the last action. Or, a tutor could accept all types of actions. Because the solution template's expectations might have been invalidated by an inappropriate action, however, the tutor would not be able to assess the subsequent actions reliably. However, as long as the simulation is able to behave realistically in response to subsequent actions, this instructional approach still gives trainees an opportunity to realize their mistake and experience their effects. For example, experiencing the simulated loss of a satellite due to operator error can be a motivating and memorable learning experience. Afterwards, the tutor could ask questions that prompt the trainee to reflect on his or her actions to figure out when the error was made, what the correct action should have been, and what the impact of the error was on the satellite or ground systems.

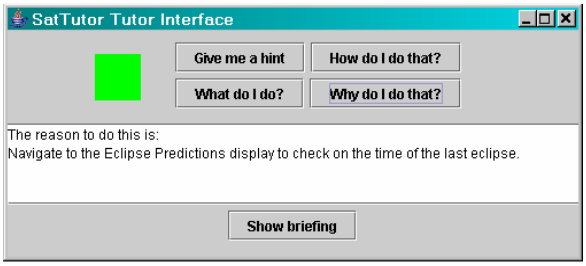

Figure 1. The tutoring system enables the trainee to ask for context-sensitive hints during exercises

D. Task Representations for Tutoring

A key design issue for any tutoring system is the manner in which task knowledge is represented, or encoded, in a computable format that can be interpreted by the software. The task representation must be expressive enough to enable the tutor to assess each action and distinguish appropriate actions from inappropriate ones, even when there is more than one correct set of actions for a given scenario. The representation must also enable the tutoring system to assess the trainee's knowledge and skills and provide useful coaching and feedback during and after each exercise. Finally, the representation must enable rapid and intuitive knowledge entry by subject matter experts so that tutoring scenarios can be created easily and economically, without complex programming.

We chose to encode each solution template as a hierarchy of simple task nodes and group task nodes that represent the set of possible sequences of trainee actions that are appropriate for a scenario. Each simple task node recognizes a correct trainee action. It specifies:

- an action pattern that specifies the action type and constraints on its parameters. An action is expected (and appropriate) if its type matches that action pattern's type and its parameters satisfy the action pattern's constraints.
- an optional simulation state condition that specifies constraints on the values of simulation state variables that must be satisfied in order for the task node's action to be active and enabled for matching against incoming trainee actions.
- optional principles (typically, specific skills or pieces of knowledge) that are demonstrated when the trainee carries out an action that matches the action pattern when the node's simulation state condition is satisfied, and

- optional text strings that are displayed when the trainee requests the various types of hints associated with each step.

Each group task node contains:

- one or more simple task nodes and/or lower level group task nodes, and
- zero or more principles that are demonstrated when the trainee carries out all of the actions that are recognized by the simple task nodes and sub-group task nodes in the group.

E. Demonstrating, Generalizing, and Annotating Tutoring Scenarios

Instructors and subject matter experts (scenario authors) use the simulator to first demonstrate one (of possibly many) correct sequence of actions for the scenario. The tutoring scenario editor records these actions to create an initial solution template that recognizes this exact set of actions performed in order.

Scenario authors then use the tutoring scenario editor to generalize this solution template so that it recognizes other valid sequences of actions. For example, the author can relax constraints on the action's parameters by specifying multiple valid values or ranges of numeric values. The author can relax ordering constraints by specifying that the actions in a group of actions can be carried out in any order. Or, the author can specify alternate sub-sequences of actions within a solution template. This feature enables the tutoring system to determine when the trainee carries out one of the several possible ways of performing a task within a scenario. Authors can also specify conditional actions that are appropriate only when certain simulation state conditions are true, expressed as a Boolean expression that refers to simulation state variables and, optionally, the action's parameters.

Authors then annotate the solution template by associating principles with actions or groups of actions. This enables the tutoring system to assign credit to the trainee for principles he or she appears to know when the action or group of actions is carried out.

III. Scenario-Specific Simulation

We developed a software framework for rapidly developing partial, scenario-specific simulations of the mission operations software, the ground station hardware and software and the satellite. The simulations are partial in that they only implement the parts of the simulated software's graphical user interface (GUI) that are relevant to each scenario. Screenshots of the actual mission operations software provide a realistic look, and interactive controls are overlaid on the screenshots only for those GUI controls that will be acted upon by the trainee during the scenario or that must display scenario-specific data that changes over the course of the scenario.

In general, it is costly and difficult to specify how a simulation of a complex system behaves in response to arbitrary trainee actions and other events. Our system avoids this problem by employing scenario-specific simulation behavior models that are valid only within a narrower envelope of the situations that are likely to occur during a given scenario, rather than any possible action or event. This approach makes it possible to quickly create scenario-specific simulations that respond realistically to those actions the trainee is likely to perform. The actions include correct actions as well as incorrect actions that are common or can be anticipated.

The models allow for parallelism and modularity by allowing the authors to create multiple entities and associating behaviors with each entity. The entities can communicate via message-passing as well as by having access to a common set of system variables.

The process of creating these scenario-specific simulation models is further facilitated by use of a behavior modeling system based on hierarchical Behavior Transition Networks. These networks have a natural visual presentation as node-and-arc graphs, and we provide a graphical editor that enables scenario authors to quickly specify simulation behaviors by direct manipulation of this visual representation. .The system also provides a debugger that enables the authors to observe and single-step through their models using the visual representation.

The underlying simulation system is called SimBionic™. We developed this simulator to enable training systems and gaming systems to simulate the behaviors of simulated agents, such as enemy forces. However, it works just as well for simulating engineered systems.

Finally, the simulator provides a mechanism for integrating externally-defined simulation codes. This allows the training simulator to take advantage of existing engineering simulations, which are generally partial simulations focused on one or a few aspects of a system's overall behavior. The simulator can import Java classes and make their methods available to the embedded expression language. Java Native Interface (JNI) can be used to enable integration of simulation codes written in other languages.

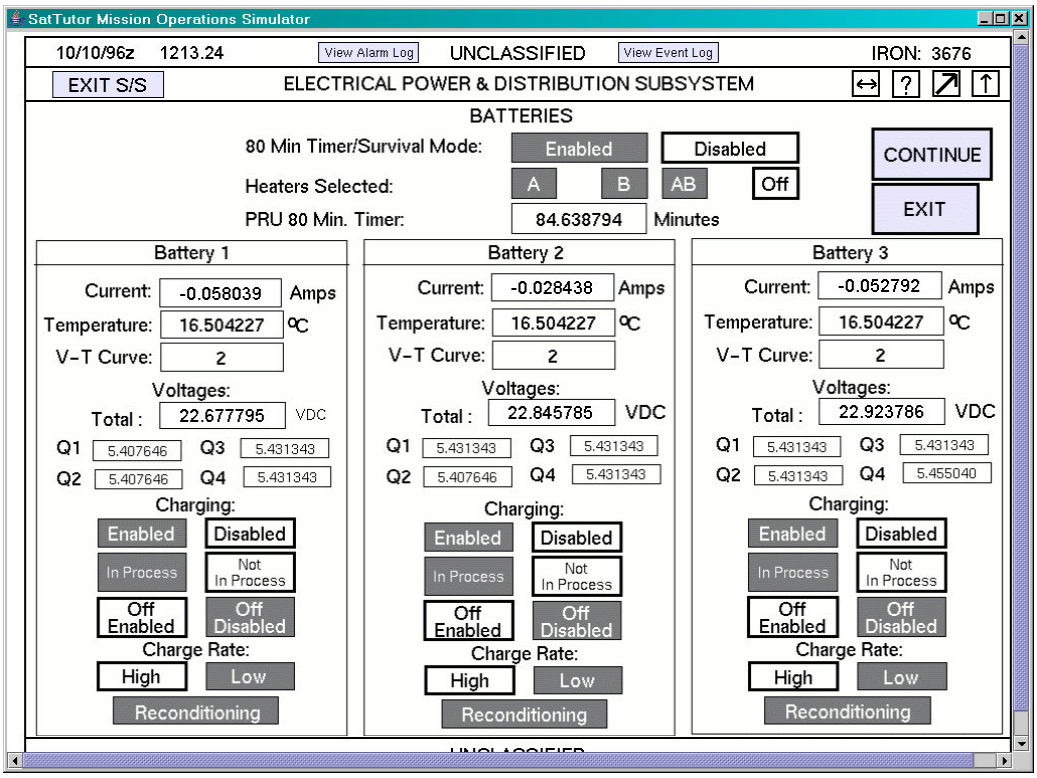

Figure 2. Rapid development of scenario-specific simulations is enabled by using screen captures of the satellite operations system's user interface and selective implementation and overlay of the user interface controls that are likely to be used during the scenario.

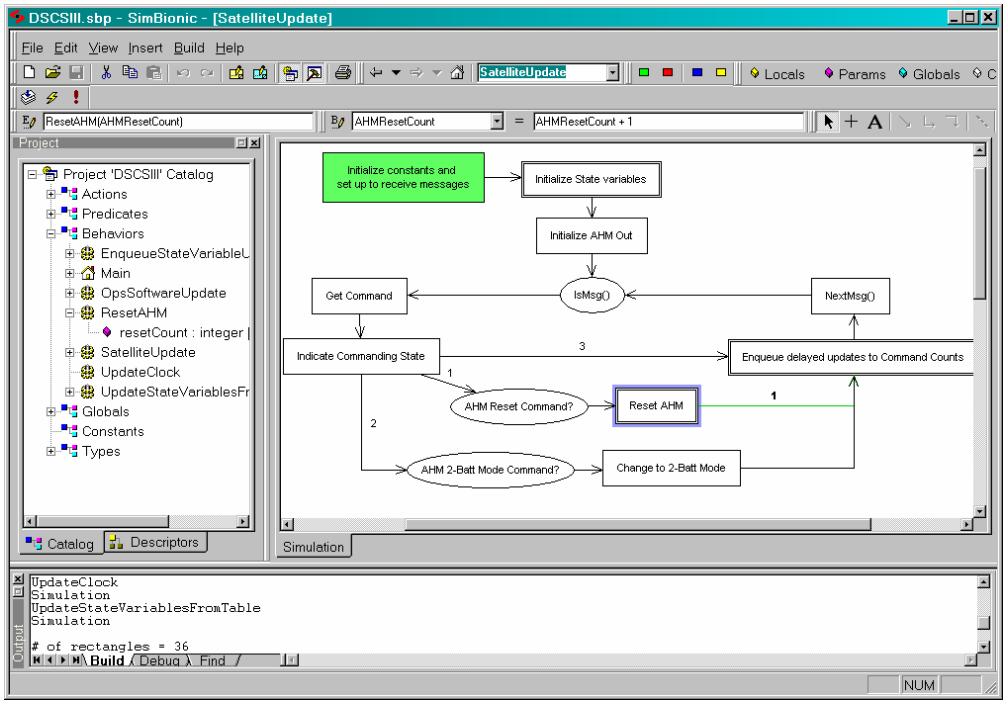

Figure 3. Simulation behaviors are defined using a graphical user interface to draw hierarchies of behavior transition networks for simulated entities.

IV. Integrated Job-Aiding and Scenario-Based Tutoring

We developed an integrated training system that combines a satellite operations simulator (developed using our TaskSim rapid scenario-specific development environment), the tutoring system (developed using the Task Tutor Toolkit), and a job-aid (TaskGuide). The job-aid supports satellite operators by giving them step-by-step instructions for carrying out satellite command plans. It accepts inputs from the operator and automatically determines the appropriate path through the command plan, which can have branching and looping. As shown in Figure 4, graphical editors enable entry and editing of tutoring scenarios and procedure specifications.

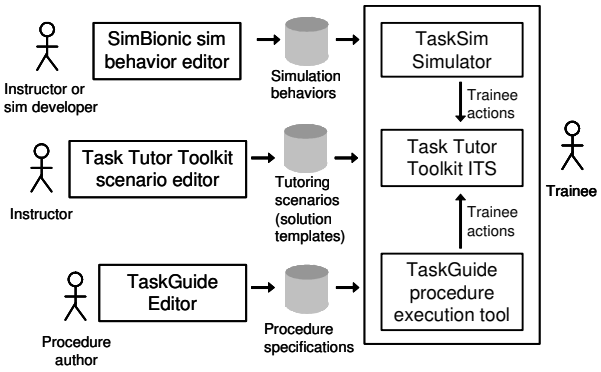

Figure 4. Integrated simulation-based tutor and job aid data flow.

V. PRELIMINARY EVALUATION AND FUTURE WORK

The system was presented and demonstrated to 10 satellite operations instructors at Vandenberg AFB in February 2005. The reaction of the participants to the software was generally positive. During the presentation and demonstration, the instructors identified enhancements to the software that they felt were the most important for acceptance of the software for operations and training.

Six participants filled out an evaluation questionnaire comprised of 22 questions that prompted each respondent to rate the usefulness or usability of various aspects of the knowledge editors and run-time systems for the job aid, tutoring system, and training simulation. The average rating for all questions and respondents was 3.9 on a scale of 1 (hard to use, not effective or intuitive) to 5 (easy to use, very effective or intuitive). 98% of the ratings were between 3 and 5. Average ratings across the three systems were comparable, ranging from 3.65 for the simulation development tool, 3.88 for the job aid, and 4.08 for the tutoring system.

The questionnaires also prompted the respondents for open-ended comments regarding the most-useful/beneficial features, the most needed enhancements, and barriers to operational use. Most comments were positive regarding the software's capabilities and ease-of-use.

We have identified potentially useful enhancements to the tutoring system. Currently, if the trainee carries out an unexpected action that is not benign, the solution template can no longer be assumed to accurately represent the next possible actions that the trainee should carry out. This is because non-benign unexpected actions may have altered the state of the world in a way that renders the solution template invalid. However, in many cases, it is possible to recover from an unexpected action by carrying out one or more additional actions that restore the state of the world so that procedure execution can proceed. To support recovery from unexpected actions that are not benign, it would be desirable to enhance the tutoring system to support *recoverable* actions. When the trainee performs a recognized recoverable action, the tutoring system can inform the trainee and guide him or her through a set of steps that recover from this action. This feature would enhance the realism and naturalness of the simulation-based exercise.

VI. RELATED WORK

Studies show that individualized instruction provided by intelligent tutoring systems are highly effective. However, a barrier to their widespread use is the cost and difficulty of encoding the subject matter and instructional expertise used by the tutoring software, especially when "deep" representations of the task are used, such as full-blown planning-style representations[3,4], and cognitive models in production-system formats[5] that enable the tutor to act as an expert system in the task area. Scenario-specific task representations avoid the complexity and expertise needed to build an expert system[6]. Authoring specific scenarios allows for focus on situations and decision points that are judged to be particularly important, and for highly tuned trainee assessment and instructional interventions. For example, Guralnik[7] describes an authoring tool that applies a content theory of procedural task knowledge, enabling the tutoring system to generate replies to important questions from the trainee. The work described in this paper builds upon prior work in software tutors for procedural training by us[2] and others[7]. Specifically, we enhanced the expressiveness of the task representations used by the tutor with constructs such as conditional actions, alternate actions, and continuable actions while striving to keep the task representations simple enough to be authored by non-programmers using graphical tutoring scenario editors.

VII. OTHER APPLICATIONS

This tutoring system and job aid can also be used to provide training and performance support for other technical tasks in which the number of appropriate ways of carrying out each task is limited. For example, these systems can help maintenance technicians diagnose and repair equipment, and they can help people operate equipment, use software applications, or perform tasks in compliance with organizational guidelines and procedures.

ACKNOWLEDGMENTS

This research was supported in part by Air Force Research Laboratory contract F33615-02-C-6063.

REFERENCES

[1]Schank, R., "What We Learn When We Learn by Doing," Technical Report no. 60, Institute of Learning Sciences, Illinois, 1995.

[2]Ong, J., S. Noneman, "Intelligent Tutoring Systems for Procedural Task Training of Remote Payload Operations at NASA," *Industry/Interservice, Training, Simulation & Education Conference,* I/ITSEC, 2000.

[3]Sacerdoti, E.D., *A Structure for Plans and Behavior*, American Elsevier, New York, 1977.

[4]Rickel, J., Ganeshan, R., Rich, C., Sidner, C.L., & Lesh, N., "Task-Oriented Tutorial Dialog: Issues and Agents," Mitsubishi Electric Research Laboratories Technical Report TR-2000-37, 2000.

[5]Anderson, J.R., Boyle, C.F., Corbett, A. T., & Lewis, M.L., "Cognitive Modeling and Intelligent Tutoring," *Artificial Intelligence,* Vol. 42, No. 1, 1990, pp 7–49.

[6]Murray, T., "Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Trainee Model, and Interface Design," *Journal of the Learning Sciences*, Vol. 7, No. 1, 1998.

[7]Guralnik, D., "An Authoring Tool for Procedural-Task Training," Ph.D. Dissertation, Technical Report #71, The Institute for the Learning Sciences, Northwestern University, 1996.