APPLYING A GENERIC INTELLIGENT TUTORING SYSTEM (ITS) AUTHORING TOOL TO SPECIFIC MILITARY DOMAINS

Dick Stottler, Daniel Fu, and Sowmya Ramachandran
Stottler Henke Associates, Inc.
San Mateo, California

and

Terresa Jackson
Air Force Research Laboratory
Brooks AFB, TX

ABSTRACT

This paper describes our experience in applying a generic Intelligent Tutoring System (ITS) authoring tool to specific training applications. The Internet ITS Authoring Tool (IITSAT) was developed to decrease the time to develop tactical decision-making ITSs and was based on the experience from several previous ITS projects. IITSAT allows ITS authors to organize course principles, articulate teaching methods, specify courseware, and develop a case base of scenarios for students along with a specification of how the student's actions will be evaluated and his mastery of the required knowledge assessed. Evaluation of the correctness of actions and inference of the student's knowledge may be performed by external code, or with libraries supplied with IITSAT. They support both the use of finite state machines (FSMs) to evaluate a student's actions in a free play simulation, or comparison to correct and likely incorrect solutions for each scenario. Different instructional methods can be chosen including who should control the sequence of instructional events - the student, the author, or the ITS, and what that sequence should be. The FBCB2/Tactical Decision-Making ITS prototype teaches armor company commanders by presenting course material and examples, then testing the commander in tactical situations displayed as FBCB2 overlays or in a commercial tank simulator interfaced to the actual FBCB2 software and the ITS. IITSAT's comparison libraries successfully evaluated a student's battle plan with the addition of domain-specific code. The next ITS to be developed with IITSAT was an F/A-18 Air Tactics ITS prototype which intelligently evaluated a pilot's actions during mission rehearsal to practice perishable skills. IITSAT was interfaced to ACM, a commercially available flight simulator which was altered to output a log of actions and events. FSMs evaluated the correctness of the pilot's actions and inferred mastery of different principles. The Air Tactics ITS was developed in a small fraction of the normal time and IITSAT did not need to be modified, but FSMs were less general than planned. An authoring tool was very helpful since it could be modified to increase its generality and flexibility.

ABOUT THE AUTHORS

Dick Stottler co-founded Stottler Henke Associates, Inc. (SHAI) in 1988 and is the president of the company. He has been principal investigator on a number of intelligent tutoring system projects.

Dan Fu is an AI Researcher and project manager at SHAI. Dr. Fu currently leads the IITSAT project and an Immersive Wargaming project.

Sowmya Ramachandran is an AI Researcher at SHAI with a strong background in a wide variety of Artificial Intelligence techniques, including Intelligent Tutoring Systems and Machine Learning.

Terresa Jackson is a program manager with the Air Force Research Laboratory at Brooks AFB, TX. She manages contractor, military and civilian research in design and development of advanced training technologies, intelligent training systems, and distance learning techniques.

APPLYING A GENERIC INTELLIGENT TUTORING SYSTEM (ITS) AUTHORING TOOL TO SPECIFIC MILITARY DOMAINS

Dick Stottler, Daniel Fu, and Sowmya Ramachandran
Stottler Henke Associates, Inc.
San Mateo, California

and

Terresa Jackson
Air Force Research Laboratory
Brooks AFB, TX

INTRODUCTION

This paper describes our experience in applying a generic Intelligent Tutoring System (ITS) authoring tool to specific training applications. We will first describe that tool and the ITS projects which contributed to its development. We will then discuss the applications that were developed using the tool. For each application we will describe its functionality and the process and benefits of using the tool as well as the difficulties. Finally we will summarize the lessons we have learned in applying the generic tool to several ITS applications and discuss the future work.

IITSAT DESCRIPTION

IITSAT was developed specifically to greatly facilitate the development of ITSs for tactical decision-making. Tactical instructors universally agree that coached practice of decision-making in tactical situations is most important for development of a high-level of expertise in tactical decision-making [Lussier IITSEC 2000 ref]. Therefore, IITSAT was designed to provide instruction centered around scenarios (i.e. cases) and required principles.

In case-based ITSs each case (or scenario) should include (1) a multi-media description of the problem, which may evolve over time (as in tactical simulated scenarios); (2) a description of the correct actions to take, possibly including order-independent, optional, and alternative steps; (3) multi-media explanations of why these steps are correct; (4) the list of methods which determine whether the steps have been correctly executed by the student; and (5) the list of principles required to know the correct action to take, typically extracted from the explanations that accompany the solution steps.

Cases for Correct Action Determination

The most difficult and domain dependent aspect of the ITS (after the simulation itself) is the determination of the correctness or incorrectness of a student's action. Since there are domains where it is impractical to build a general expert system to produce the correct actions, the expert's knowledge of the correct actions specific to a scenario are stored within the scenario itself. This knowledge typically takes different forms, based on the domain and the ability of the student to alter the flow of the scenario in unexpected or multiple ways. The simplest representation lists the correct actions at the appropriate time in the scenario. Obviously this will only be applicable if the flow of the scenario is unaltered by actions of the student or if at each mistake, the student is immediately corrected, and thus the scenario's timeline is restored. For each scenario, methods are required for comparing these correct actions to the actual actions produced by the student. These methods may also be able to assess which principles associated with a particular action the student knows and which ones he doesn't, based on a whole or partially correct action. For example, in some AWACS Weapon Director (WD) scenarios, the WDs are supposed to advise rather than command. Thus the scenarios can be structured such that the simulated pilots ignore WD mistakes, and the scenario timeline proceeds unaltered. The WD actions are the advice, specific utterances made to specific pilots over the simulated radio, usually less than 20 words each. The correct actions are the utterances of expert WDs, previously recorded while they played the scenario. The software methods to compare the correct actions to the student's actions must convert each to a text representation. The WDs, according to their orders, are supposed to use a specific grammar. This allows the text to be parsed and compared piece by piece. The software methods can then assign knowledge of principles based on subparts of the student's utterance. Some principles, such as "give the most important information first," actually span multiple actions, as well.

Of course these types of scripted scenarios preclude one of the most important learning opportunities - for students to see the results of their own mistakes. Mistakes a WD makes in real missions can easily cause loss of life, including his own. So there is a strong desire to use more flexible and dynamic simulations and scenarios, where a student's actions can radically affect the outcome. Since these simulations are typically continuous, there are an infinite number of variations that different students can create. In fact, in these types of situations the same scenario never plays exactly the same way twice, since minor timing differences of student actions affect the precise trajectories of the simulated players. Clearly, listing the correct action at the appropriate time, based on the way the expert played the scenario is inappropriate, since when the student plays the same scenario, his timeline will diverge from the expert's, often in radically different ways. For example, a particular scenario may dictate that the student remain covert while gathering information. If he understands how to do this, the enemy may never detect his existence, and thus never attack him. However, a student who does not understand the principles of covertness may turn on his active sensors, be detected by the enemy, and thus come under attack. At this point he may correctly assess the need for and execute several self-defense actions. These actions were not required of the expert or of other students in the same scenario who performed the information gathering tasks in the correct, covert way. Yet, they are entirely appropriate for the situation in which the student finds himself, and not only should they not be considered incorrect, but he should also get credit for understanding the appropriate self-defense principles.

The solution is the other extreme of the forms of knowledge, in which knowledge of correct actions may be stored and used is in situations where the system in no way can produce the correct or all the possible correct actions but for which the knowledge exists, within the context of a scenario, to evaluate the appropriateness of the student's action. For example, to refine the location of an enemy platform, an aircraft may be sent to a general area. To keep the aircraft's home platform location unknown, it should take an indirect route to the area. There may be several factors to consider when determining an appropriate route, many of which may be considered commonsensical or at least not part of the course the ITS is teaching. The ITS may not include the knowledge required to generate a good route. Furthermore, there may be a very large number of acceptable routes. But, for the purposes of making sure that the student understands the concept of taking an indirect route to the target area, it is fairly easy to devise a simple calculation to check that the route was indirect. One way to represent these types of scenario specific evaluation machines is using Finite State Machines (FSMs) which are discussed later. (Figure 1) shows a FSM used for evaluation in an ITS prototype developed with IITSAT.

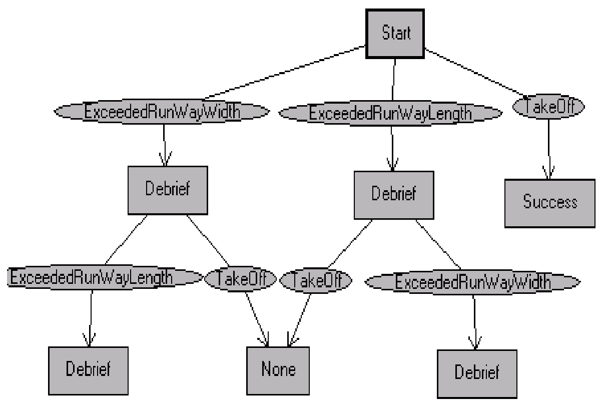
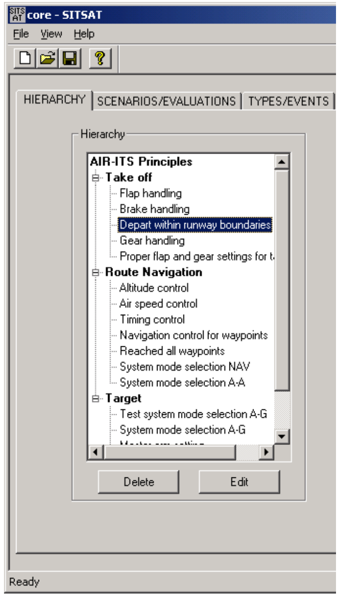


Figure 1. Example Finite State Machine.

Creating an ITS with IITSAT

There are six kinds of knowledge that must be entered by the domain and/or instructional expert to create an ITS for a specific domain. These are the case base of scenarios to be used as examples and exercises, the hierarchy of principles referenced from those scenarios, multi-media descriptions which explain each principle, knowledge used to asses the correctness of student actions, knowledge used to assess a student's mastery of a principle given the history of his performance in relation to that principle, and pedagogical knowledge. Methods to enter the scenarios tend to be very domain specific and closely tied to the simulation. For tactical scenarios, typically graphical editors are employed based on an electronic map and intelligent tactical knowledge entry techniques not particular to ITS concerns. The principle hierarchy is entered through a simple tree-based graphical editor as shown in Figure 2. The multi-media descriptions of principles are entered using commercial multi-media authoring tools, such as Macromedia Director. The different ways to represent the knowledge to determine student action correctness was discussed in a previous section. In the following paragraphs we discuss the capabilities for entering mastery assessment and pedagogical knowledge.

Representation and entry of the knowledge to assess principle mastery, given a history of actions related to it, is one of the simplest aspects of the authoring tool. The author specifies and names the levels of mastery. For example, those might be novice, intermediate, and expert. For any principle in the hierarchy, he then defines the conditions that must be met to attain each level of mastery. These conditions typically define the percentage of correct usage of a principle from the last N actions using the principle in the last M scenarios in a specified time period. The required parameters are simply entered using a fill-in-the-blank format. Which principles the mastery level applies to is determined by which principle node the author selected in the principle hierarchy editor. The mastery assessment definitions defined at a higher level in the hierarchy are inherited by all of its subprinciples unless over-ridden with a more local definition.



AIR-ITS Principles
- Take off
  - Flap handling
  - Brake handling
  - Depart within runway boundaries
  - Gear handling
  - Proper flap and gear settings for t
- Route Navigation
  - Altitude control
  - Air speed control
  - Timing control
  - Navigation control for waypoints
  - Reached all waypoints
  - System mode selection NAV
  - System mode selection A-A
- Target
  - Test system mode selection A-G
  - System mode selection A-G

Figure 2. Principle Hierarchy Editor.

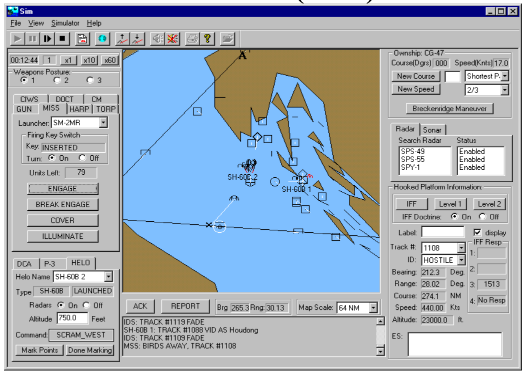
More complicated is the pedagogical knowledge. A somewhat simplified description follows. The authoring tool allows different instructional methods to be defined for different types of students (based on background and principle mastery) and different regions of the principle hierarchy. Aspects of an instructional method include degree of instructional support; degree of student control; how much instructional material to present; what kinds of examples to show, and how many; what kinds of exercises to present, and how many; type and timing of debriefing; remediation, and exercise selection.

Perhaps the greatest challenge in designing the functionality and capabilities of the ITS authoring tool was maintaining the proper balance between flexibility/power and usability. By designing a lot of flexibility in the instructional method specification, many inputs are required. Our design philosophy is based on the assumption that domain knowledge experts (a.k.a. Subject Matter Experts) are more readily available than pedagogical experts and that pedagogical knowledge can be generalized over domains. We therefore have made pedagogical knowledge specification easy by having the authoring tool intelligently select default specifications that an author can choose to over-ride. We are developing a case base of instructional techniques so that when some preliminary information about the domain and types of students is entered, the system selects the most appropriate default instructional techniques for each type of student and principle. A Subject Matter Expert is able to generate an ITS by just specifying domain-specific knowledge (principles, scenarios, pre-test and post-test scenarios), and using default specifications for pedagogical knowledge.

ORIGINATING APPLICATIONS

IITSAT was designed and implemented based on the experience of developing several tactical decision-making ITSs. Each included certain methods and techniques which contributed to IITSAT. These are described below. The TAO ITS is described in more detail, since, as described at the end, the newest fleet version is being converted to IITSAT.

Tactical Action Officer (TAO) ITS



00:12:44 1
Weapons Posture: [✓ 1] 2 3
Launcher: SM-2MR
Firing Key Switch
Key: INSERTED
Turn: [✓ On] Off
Units Left: 79
Helo Name: SH-60B 2
Type: SH-60B LAUNCHED
Radars: [✓ On] Off
Altitude: 750.0 Feet
Command: SCRAM_WEST
Brg 265.3 Rng 30.13
Map Scale: 64 NM

```
IDS: TRACK #1119 FADE
SH-60B 1: TRACK #1088 VID AS Houdong
IDS: TRACK #1109 FADE
MSS: BIRDS AWAY, TRACK #1108
```

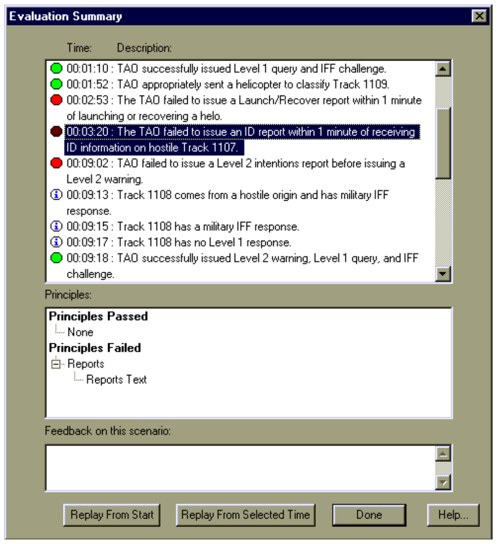
Ownship: CG-47
Course(Dgrs) 000 Speed(Knts) 17.0
Shortest P.
2/3

Search Radar	Status
SPS-49	Enabled
SPS-55	Enabled
SPY-1	Enabled

Hooked Platform Information:
IFF Doctrine: [✓ On] Off
Label: [✓ display]
Track #: 1108
ID: HOSTILE
Bearing: 212.3 Deg.
Range: 28.02 Deg.
Course: 274.1 NM
Speed: 440.00 Kts
Altitude: 23000.0 ft.
IFF Resp
1:
2:
3: 1513
4: No Resp

Figure 3. TAO ITS Simulation.

SHAI designed and built for the Surface Warfare Officers School (SWOS) a low-cost simulation-based intelligent tutoring system (ITS) for use on standard PCs as part of SWOS's Tactical Action Officer (TAO) training program to train Navy officers in high-level tactical skills in early 1999. The TAO ITS Simulation interface is shown in (Figure 3). A key objective of the software is to increase the active training that officers receive to improve their ability to apply their conceptual knowledge of tactics. The intelligent tutoring system presents selected scenarios for the student to practice different tactical concepts. The software adaptively selects scenarios for individual students that practice concepts he or she hasn't yet practiced or has recently failed. As well as the intrinsic feedback that free-play simulations naturally provide a student, the TAO ITS provides detailed, useful extrinsic feedback to the student once a scenario is finished, which reviews the student's decisions along with the related concepts and whether they were passed or failed (as shown in Figure 4). At this point, the student can review multimedia material about any concept, or see a replay of the scenario to review errors.



● 00:01:10 : TAO successfully issued Level 1 query and IFF challenge.
● 00:01:52 : TAO appropriately sent a helicopter to classify Track 1109.
● 00:02:53 : The TAO failed to issue a Launch/Recover report within 1 minute of launching or recovering a helo.
● 00:03:20 : The TAO failed to issue an ID report within 1 minute of receiving ID information on hostile Track 1107.
● 00:09:02 : TAO failed to issue a Level 2 intentions report before issuing a Level 2 warning.
ⓘ 00:09:13 : Track 1108 comes from a hostile origin and has military IFF response.
ⓘ 00:09:15 : Track 1108 has a military IFF response.
ⓘ 00:09:17 : Track 1108 has no Level 1 response.
● 00:09:18 : TAO successfully issued Level 2 warning, Level 1 query, and IFF challenge.

Principles Passed
- None

Principles Failed
- Reports
  - Reports Text

Figure 4. TAO ITS Debriefing.

TAO ITS follows a scenario-debrief instructional cycle. That is, it selects a scenario that it believes would be beneficial for the student, has him perform tactical decision-making in that simulated scenario, then debriefs him on the correctness of his actions. It also provides information on the concepts that it feels he is deficient in, based on the mistakes he just made. IITSAT was designed to include this type of instructional method in its ITSs. But as we transitioned the TAO ITS for use in the fleet, we found that other instructional cycles would be required for refresher training with less knowledgeable students. Therefore, IITSAT also allows for the specification of different instructional methods for different types of students and these include introduction of new material and presentation of examples, before the student is forced to perform in a simulated scenario. TAO ITS communicated to its simulation through an event log file, which was analyzed using Finite State Machines (FSMs) definable by instructors. TAO ITS was designed this way, since early in the project interfacing to as-yet unspecified simulations was deemed important and a log file interface make this very simple.

Furthermore, the restriction of feedback to the student occurring only after the end of the scenario was not considered important since the debrief, or After Action Review (AAR), was considered to be the primary feedback mechanism. The FSM evaluations proved so successful that TAO ITS's FSM code was incorporated, with only minor modifications into IITSAT.

Other ITSs

An Intelligent Tutoring System was developed to teach the principles and processes of sonar image analysis. The ITS complements the existing interactive courseware by providing practice in simulated acoustic analysis scenarios, with an automatic debriefing capability. The ITS models the student's knowledge and abilities and selects the most appropriate practice scenarios for each student. The scenarios are created though an annotation authoring process by expert acoustic analysis instructors. This ITS contributed its Principle Hierarchy editor to IITSAT as well as the concept that the scenario-player may not always be a tactical simulation. Its scenario player is an annotation editor which allows the student to annotate an image, choose different processing options (which are like different "views" of the same data,) and which only provides access to the data that would have arrived at the particular point in time.

An ITS was developed for the Army's Military Intelligence Training Distance Learning Office at Fort Huachuca which uses a constructivist approach to teach principles of intelligence analysis for countering terrorism. This project contributed the concepts of the importance of hinting and coaching during scenario play; the need to specify specific scenarios for specific parts of the course; the fact that the number of scenarios may be very small in number; and the requirements of pre and post testing.

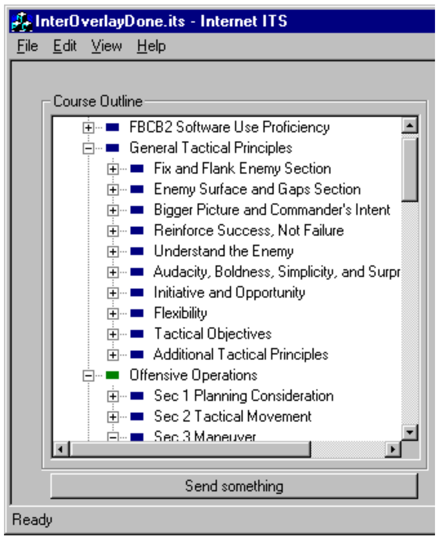
APPLICATIONS CREATED WITH IITSAT

FBCB2/Tactical Decision-Making ITS

The FBCB2/Tactical Decision-Making ITS teaches the tactical use of FBCB2, an Army C4I system, and tactical decision-making to Armor and mechanized Infantry company commanders. When a new student logs on he is first asked some questions about his background, experience, and last FBCB2 training/use. These questions include level of education achieved, rank, highest unit commanded, types of units served in, computer familiarity, BCB2 familiarity/comfort, and general perceptions as to its usefulness. The ITS uses this information to make initial estimates as to the student's mastery of various principles, including both tactical knowledge and the use of FBCB2. It is also used to select scenarios, other exercises, types of hints, and other forms of instruction. Mastery categories are Beginner, Novice, Intermediate, and Expert. The Beginner category for a principle occurs when a student performs successfully with it less than 20% of the time. (Novice – 20 to 50%, Intermediate – 50 to75%, Expert > 75%). Students at the expert or intermediate level for a principle are never given hints.

If the ITS estimates that the student's mastery of FBCB2 principles is low, then before doing simulated exercises, the student is first put through FBCB2-only refresher exercises. An introductory lesson explains with detailed steps how to create an overlay and find and place the most relevant symbols.

After the FBCB2 refresher exercises (if they were needed), the ITS begins tutoring the student on general tactical principles. If it estimates his mastery is relatively high it proceeds immediately to tactical decision games presented and answered as FBCB2 overlays. If not, it first presents general tactical principle courseware. For each tactical decision game (TDG), the ITS analyzes the student's plan (given as an FBCB2 overlay) and automatically creates a debriefing describing what parts of his plan are right, what parts are wrong, and gives an expert's rationale for the best options. For poor decisions, the ITS lowers its estimate of the mastery of principles related to those decisions, and provides remedial materials on those principles, before presenting anymore TDGs. The student's overlay is evaluated by comparing it to overlays input by an instructor for that particular TDG. These typically represent a few possible right answers and a few common mistakes. The instructor will also have annotated the overlays with information for use by the ITS in assembling the debrief and determining which principles the student is weak in. A sample of the course hierarchy in IITSAT is shown in (Figure 5).


InterOverlayDone.its - Internet ITS

File Edit View Help

Course Outline

- FBCB2 Software Use Proficiency
- General Tactical Principles
  - Fix and Flank Enemy Section
  - Enemy Surface and Gaps Section
  - Bigger Picture and Commander's Intent
  - Reinforce Success, Not Failure
  - Understand the Enemy
  - Audacity, Boldness, Simplicity, and Surpr
  - Initiative and Opportunity
  - Flexibility
  - Tactical Objectives
  - Additional Tactical Principles
- Offensive Operations
  - Sec 1 Planning Consideration
  - Sec 2 Tactical Movement
  - Sec 3 Maneuver

Send something

Ready

Figure 5. IITSAT's Student Interface.

For the TDGs and the 3-D dynamic scenarios, the ITS initially selects exercises based on the need to test untested principles, following each by a debriefing and detailed information on the principles missed. The ITS then begins to also retrieve scenarios that exercise the principles in which the student's mastery is weakest. Furthermore, for any scenario using principles that the ITS believes the student is weak in, it provides him hints for the scenario, if they are available. These are generally questions designed to get him to think about the most important tactical principles required in the scenario.

After the student has demonstrated (or learned) his mastery of general tactical principles in the TDGs, he proceeds to that portion of the course that requires him to show that he can apply these same principles in a 3-D virtual reality dynamic tactical simulation (For this phase we used Mak Technology's Spearhead II, shown in Figure 6). Additionally, more operations-oriented principles (such as knowing when and how to use a company wedge formation) are also tested. In the current prototype, the student is given a short situation description and then proceeds to execute the mission in Spearhead II. After the scenario ends, the event log is analyzed by the ITS to automatically determine which actions were correct, incorrect, or omitted, and the underlying principles that were understood and applied or not.

Figure 6. Mak Technology's Spearhead II.

In some scenarios, we have subordinates that do not follow orders, plans, and proper tactics. Normally the commander would correct these problems with voice commands. In this prototype we do no speech understanding. But these corrections should be manifested by the motions and actions of the commander's company's tanks, of which he has direct control. The ITS assesses these motions and actions (captured from Variable Message Format (VMF) messages). For example the commander's OPORD may have had the lead platoon in a wedge formation but it is proceeding in a column. If he orders them into the correct formation, an evaluation machine detects the correction and he gets credit for recognizing the wrong formation, and recognizing the need to correct it. If they continue to move as a column, he fails these principles.

Some scenarios in particular test his use of FBCB2 to maintain situational awareness. For example, in one scenario the enemy is approaching from an unexpected direction, which is trivial if the commander is watching the FBCB2 map display. Another test we use is to have friendlies show up suddenly at an expected enemy location.

In the scenario, unplanned actions occur, such as unexpected contact with the enemy. His tanks begin to react and he also issues particular orders, verbally in the real world, with mouse clicks in the simulation. Again the correctness of his decisions is evaluated from the movements and actions of his company's tanks. For example, one scenario involves the lead platoon spotting a road block at a choke point. That platoon should deploy in a support by fire position and the commander should order his infantry to protect each flank. He should then order dismounted assaults up each flank and around the road block to secure the far side. He should Call For Fire at appropriate locations and times during the scenario as well. Evaluation finite state machines check each of these actions and debrief the student at the end of the scenario as well as infer the state of his tactical knowledge. A test for the combat principle of audacity is to have the commander unexpectedly come across a much larger force in a totally unprepared situation, such as refueling, without security.

After the scenario, the commander is debriefed with an After Action Review. All the things he did right and wrong are reviewed and he is told about the relevant principles. For the failed principles he is given detailed information and one example for each. The mastery level estimates for all principles involved are updated. Based on these, a new scenario is retrieved. Scenarios are selected that test untested principles and test recently failed principles. The prototype has different instructional methods for students with little mastery or experience compared to students with a lot of mastery and experience.

Process/Benefits of using IITSAT

To develop an ITS in IITSAT, first requires determining the content and target students. Any important information that the ITS should ask each student about his background is first defined. IITSAT organizes content in a book metaphor with a course consisting of chapters and these in turn consisting of sections, which is the main instructional unit. Sections are assumed to teach a set of principles. Each section has detailed and summary multimedia files associated with it, along with scenarios to use as examples. A section's principles have detailed and summary descriptions to be used during remediation (when the ITS determines that a student failed to apply a principle in a scenario). The ITS author must organize the content into principles, sections, and chapters.

The default for IITSAT ITSs is for new students to be in only 1 chapter at a time. (The ITS must estimate their mastery to be at least to a specified degree before it progresses them to the next chapter). For the FBCB2 ITS this corresponded to our needs exactly. The first chapter consisted of sections for creating and editing FBCB2 overlays and finding and placing common symbols. This was a needed prerequisite for the second chapter, since the student's answers to scenarios in that chapter would be input as FBCB2 overlays. This second chapter consisted of general tactical principles, which would be illustrated by their answers to (static) tactical decision games. The TDG scenarios all only referenced principles in Chapter 2 and since only TDG scenarios followed this convention, only TDG scenarios would be retrieved to practice and test mastery of chapter 2 principles.

Chapter 3 was intended to consist of the 3-D dynamic simulation scenarios and more operational principles (such as when and how to perform a bounding overwatch). Thus chapter 3's section covered operational principles. Its scenarios referenced both chapter 3 principles and chapter 2 (general tactical) principles. In this way, poor decisions that related to chapter 2 principles could be correctly assessed, and if the ITS assessed mastery of these principles was low enough, the student would be sent back to that part of chapter 2.

The ITS tracked each student's mastery of each principle into author-defined categories. It was decided that a "Beginner" category was needed for those with no experience or knowledge whatsoever. This category would always be presented with detailed material and examples before having to perform in any scenarios. This category would also be forced to successfully perform enough Chapter 1 FBCB2 overlay scenarios to prove mastery before continuing on. For some chapters, it was determined that a very high level of expertise should be required to "pass" them. Chapter 3 principles, for example, would not be tested again and so it needed a high standard to pass. This became the "Expert" level. Other chapters, such as Chapter 2, had principles which would continue to be practiced in later chapters, so a lower mastery level is allowed for passing. This became the "Intermediate" level. Finally, a category was needed that was between "Beginner" (knows nothing) and "Intermediate" (allowable for passing some chapters) and this became "Novice".

For the three chapters, scenarios were defined and annotated. The annotations consisted of the demonstrated principles, and the evaluation method. For chapter 1 and 2 scenarios, the evaluation method was comparison to stored correct and incorrect tactical plans in the form of FBCB2 overlays. Associated with each symbol was a list of the principles required to be applied to understand that that symbol should be at that location. Also the rationale for that symbol's selection and placement were stored in a text file. Code was written to convert both the student's and the stored plans from FBCB2's VMF format to a plain text format that was easier to work with. We then wrote code that could compare two symbols from two separate plans and assess their similarities. This was then embedded in the similarity assessment code that comes with IITSAT. This uses the symbol assessment to first determine the closest stored plan. It then uses that closest plan to create a debriefing for the student. For chapter 3, we used IITSAT's FSMs. We defined FSMs that analyzed the log file from the 3-D dynamic simulation and determined which actions were correct (and which associated principles were thus passed) and which actions were incorrect (and which associated principles were thus failed).

The prototype was then up and running and could be tested and refined while playing the roles of different kinds of students.

The primary benefit of IITSAT was greatly reduced development effort and time. Most of the ITS functionality we needed already existed in IITSAT and was readily accessible. IITSAT provided good instructional and course progression functionality. The scenario-based instructional paradigm was very natural for this domain. By utilizing the IITSAT feature of different instructional methods for different types of students (assessed primarily by background questions and mastery of principles), we were able to show a high degree of intelligence in our ITS. The two primary paradigms, which were matched both to different students and different chapters were the scenario-debrief and introduction-examples-scenario-debrief loops. The fact that IITSAT communicated with simulations (and other scenario players) through log files made the interfacing work as straight-forward as it could be. Similarly it required no effort for IITSAT to communicate the need for hints to the scenario player for beginners and novices. The hints generally took the form of a question (such as what is the enemy probably thinking?) or the advice to consider a particular principle or aspect of the scenario. The FSMs performed well in evaluating the student's actions in the dynamic simulation.

Difficulties

There were several major challenges associated with this ITS. Some were outside of IITSAT's intended scope. Getting FBCB2 running on a desktop system instead of in an actual vehicle was very difficult and time-consuming. Getting the existing interface running between FBCB2 and the commercial game was very difficult and time-consuming, since we had no budget for and therefore little cooperation from the developers of those systems. We had originally planned to interface with both the game (to get a log of events in the 3-D simulation) and FBCB2 (to get the overlays and message traffic) but we eventually had to settle with just getting the FBCB2 overlays and using the FBCB2 log of events. We had assumed that reading the overlay files would be straight-forward, but we had to acquire special decompression software to decode the Variable Message Format (VMF) in which the overlays were stored.

Although it assumes a scenario-based paradigm, IITSAT provides no simulation on which to play scenarios. Similarly it provides no scenario editor. For Chapter 3 scenarios, we used a slightly altered version of a commercial game. While this was acceptable for a proof-of-concept prototype, the game would be unacceptable for actual training use for company commanders, our target student. Additionally for the decision evaluation, we were forced to write domain specific code to serve as primitives in the FSMs and as primitives in the plan comparison.

Because IITSAT communicates to simulations via files, it pauses while the simulation is running (waiting for the log file). This means, that although IITSAT may determine that a hint is appropriate, it can only signal that fact to the simulation when the simulation is invoked and cannot execute a hinting mechanism itself. Thus we had to write a hinting mechanism into each scenario player. This turned out to be straight-forward, however, since we decided to display the hints at the beginning of the scenario. But, it would have been impossible to dynamically hint during the simulation run. This limitation will be corrected in the next version which will also include an HLA interface.

There were four other problems with IITSAT which have since been corrected. During the FBCB2 ITS development, IITSAT only allowed one executable to be defined for all scenarios. But we had two different applications (FBCB2 overlay editor for Chapter 1 and 2 scenarios and Spearhead II for Chapter 3 scenarios). For the prototype, we wrote a small executable which was defined to IITSAT to be the one and only scenario player. This application, when called with a named scenario, simply determined the correct scenario-player for the scenario from a text file entered by the author, and called it. Another quirk of that version of IITSAT was that it let the author define the initial set of background questions when the ITS was first created, but not change it. The author had to make sure every question was determined at the time the ITS was first created. Needless to say, this did mean recreating the ITS a number of times. Fortunately, this process is not time-consuming since it only involves entering the names of scenarios, principles and multimedia descriptive files, not recreating them. Another limitation of the background questions is that they could only be used to assess an average mastery for all principles, not a specific mastery for a specific principle. Finally, the version of IITSAT that we were using, instead of allowing the student to enter the authored ITS with a single mouse click, required the student to start up IITSAT, load the correct course, load the student model that corresponded to himself, and, when running the first scenario, select the scenario-player executable (which IITSAT would remember thereafter).

Because IITSAT is based on a scenario-based instruction paradigm and because much of its adaptability is manifested in an intelligent selection of the best scenario for a specific student, it works best if a lot of scenarios are defined. This can be very inconvenient during the development of a prototype ITS or the initial stages of an operational ITS. During the course of developing the FBCB2 ITS, we made minor adjustments to IITSAT's scenario retrieval algorithm to improve its use of a limited number of scenarios. But it was still the case that adding many more scenarios would improve its adaptability.

Another IITSAT change made during this ITS development greatly increased the efficiency of its XML format storage of courses and student models.

The use of FSMs for student action evaluation required that each FSM machine had to be tied to the specific terrain in each scenario. Any transition in the FSM that referred to a location had to specify that location as Lat/Long coordinates. For example, in the scenario where the company was proceeding along a road and encounters a road block, the transition that checks that a mechanized infantry platoons is deployed to the left flank actually checks that the location of one of them is at a particular lat/long location, within a tolerance distance. This limited the types of scenarios that could be practically handled. Our next version will dynamically calculate and use terrain features, such as ridges, hills, valleys, and intervisibility lines.

The use of IITSAT was straight-forward when the defaults were acceptable. But there was a steep learning curve associated with taking advantage of the more advanced features when the default behavior was not desired. It took some time to understand what IITSAT was doing and why after the defaults were changed (though it always turned out to be behaving correctly). For example IITSAT's defaults specify that students should finish one chapter before being able to select another. While this makes sense for less capable students, it will tend to frustrate more knowledgeable ones. IITSAT does allow different instructional methods for different types of situations, but when certain students are allowed to select multiple chapters, there are other more-subtle consequences. In general, working out these types of control issues can cause unexpected (but correct) behavior. The control issues relate to how much freedom of choice the student has as to the next instructional event compared to the control exercised by the ITS to dynamically force specific instructional events in a certain order, compared to the author statically defining what that order should be.

One last difficulty related to the fact that we were developing a prototype ITS primarily for demonstration. IITSAT was designed primarily to develop actual operational ITSs. Making the same choices in defining a demonstration prototype that would have been made in developing the operational ITS results in a prototype that requires a very long demonstration. For example, typical scenarios in the prototype require from 10 to 20 minutes. Five or six are required to get though Chapter 1; from 4 to 6 to get through Chapter 2, and several to at least illustrate Chapter 3. More scenarios are required if performance in the scenarios is poor. And to illustrate the adaptability of an ITS for different students, generally requiring viewing its decisions on at least two different students. This requires about 8 hours of demonstration! A useful capability would be to define a parallel, demonstration version of many of IITSAT's parameters.

F/A-18 Air Tactics ITS

IITSAT was used to develop a prototype air tactical intelligent tutoring system that provides pilots with instructional feedback automatically, allowing the pilot to identify and concentrate on perishable skills. The prototype was based on a cognitive task analysis for F/A-18 missions, completed with the assistance of a subject matter expert. A complete system consists of a simulator, evaluator, training system, and mission planner. The prototype comprises the evaluator and training system interfaced to a commercial flight simulator. The use of IITSAT made the development of the ITS prototype, within a very limited budget, possible. The graphical Principle Hierarchy editor allowed the domain knowledge to be defined with very little effort and was able to model the F/A-18 Air Tactics knowledge adequately. IITSAT's student model definitions were adequate for modeling pilots and required very little time. Changes could be made easily. IITSAT's instructional methods structure did allow the intelligence to generate the needed sequence of instructional events to be defined with little effort. IITSAT provided the visual tools to aid the authoring of “evaluation machines” that assess pilot performance. Several machines work in concert by taking a simulator log of events as input and producing a debriefing report for the pilot and tutoring system.

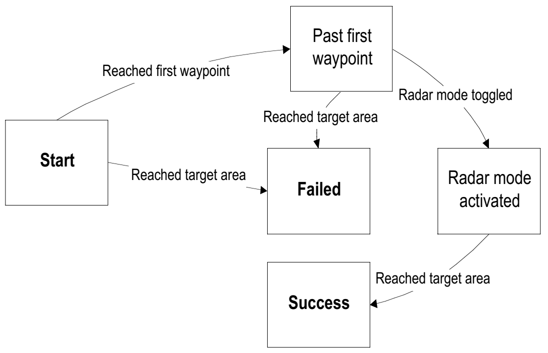


Figure 7. An F/A Air Tactics Finite State Machine.

IITSAT's evaluation machine technology was adequate for most purposes. It included visual tools for creating a set of mission-related principles with an associated twenty-five evaluation machines. Visual tools greatly improved the efficiency of authoring evaluation machines. (Figure 7) illustrates such a machine. See that the only way for the machine to reach the “Success” state is for the pilot to reach the first waypoint, toggle radar modes, and then reach the target area. Should the pilot forget to toggle radar modes, miss the first waypoint, or toggle radar modes before the first waypoint, the “Failed” state will be reached. As a result, the pilot can expect a debriefing associated with this type of evaluation. Our evaluation machine technology proved to be invaluable in producing a useful debriefing for the pilot.

The subject matter expert guided the creation of feedback output so as not to offend the pilot. Part of the technology included visual tools for viewing debriefings.
To analyze pilot performance on a mission, we first needed a content vocabulary to serve as a touchstone for basing debriefings as well as deciding the next training exercise. The vocabulary is embodied in what we refer to as the "principle hierarchy." Each principle can be a perishable skill or competency upon which we can evaluate the pilot.

Overall, the IITSAT evaluation software module proved invaluable for authoring the logic to assess performance. A time-consuming element of the evaluation machines is in the definition of the interface between the simulator and evaluation machines, as each machine must "understand" the syntax of the mission log.

Originally the development team had assumed that many of the evaluation machines would be reusable across scenarios. However, it became easier to define many evaluations machines that were specific to specific scenarios. In considering other types of more complex evaluations, a ceiling was reached because the evaluation machines are based on finite state technology. Evaluations involving pattern recognition are much harder than the simpler conditional logic implicit in the structure of the machines, although more complex pattern-matching primitives could have been defined, programmed and incorporated into the finite state machines. Certain evaluations could not be achieved because of limitations on our evaluation machine architecture. For example, a pilot may decide to skip waypoint 2, yet still achieve objectives by flying directly to waypoint IP. The evaluation machine for the principle, "Arrived at waypoint selected," will signal an unmet objective. Further, the time of arrival at waypoint IP will be earlier than expected—again an unmet objective. These specific cases could be handled by adding more links, to create paths that correspond to all correct sequences, but these could become very numerous. The full-scale, operational ITS implementation would most likely require a more powerful machine, capable of expressing more complex types of evaluation so as not to overburden the author.

Other types of evaluations are imaginable, but would push the limitations of the technology. For example, in air-to-air combat, two fighters may be scissoring, causing a pilot to stall the plane. An evaluation machine could notice the stall event happening, but not recognize the preceding scissoring motions. Either a more powerful evaluation machine is necessary, or another module is needed which can interpret the mission log and inject a high-level "scissoring motion" event prior to evaluation.

Early on, we encountered problems attaining source code or documents regarding any one F/A-18 flight simulator. Companies were unwilling to furnish source code, or had inadequate methods for transmitting or storing mission data. The simulator we eventually adopted is "ACM: air combat simulation for Unix and Windows," which is a low fidelity F-16 simulator. Information on the simulator can be found at http://www.websimulations.com. The simulator comes with source code written in C for the Windows NT platform. Having source code freed us to modify the simulation. Even though the simulator was for an F-16, it was possible to convert the simulator to support F/A-18 look-and-feel. We modified the simulator in four ways. First, the HUD was rearranged to look more like an F/A-18 HUD. Second, route information was added. The simulator shows the pilot a bearing to the next waypoint, shows the name of the waypoint, and notifies the pilot when he reaches a waypoint . Third, we added air-to-ground mode for ground attack, as well as master arm mode. Fourth, the simulator produces a mission log file so that the ITS-AIR system could determine what had happened in the simulation, and therefore evaluate pilot performance.

In summary, IITSAT's benefits for this application were greatly reduced development time since it provided a large majority of the needed functionality, the simulator log file interface was to easy to work with, and the finite state machines (FSMs) handled most assessments well and provided good, easily tailorable debriefings to pilots. The major difficulties were finding a simulation that could be altered to produce a log file, and that there were some evaluations that could not be performed adequately by the FSMs. In a full-scale system, these would require C++ programming. Lastly, many of the FSMs needed to be written to be scenario specific.

TAO ITS Fleet Transition

The TAO ITS in use at SWOS, as described earlier, was funded to transition to fleet use [Stottler and Harmon 2001] and it was determined that fleet student TAO users were more diverse than SWOS students in terms of their familiarity with the knowledge required. Specifically TAO ITS originally followed a scenario-debrief instructional cycle which is appropriate when the students are familiar with the majority of the knowledge needed to perform reasonably well in the simulated scenarios. This is to be expected in a schoolhouse environment when the material will be fresh. But many TAO students in the fleet would not have this level of knowledge either because they had not taken the SWOS TAO course or it had been too long since they had. TAO ITS needed to incorporate different instructional methods for different types of students in this more diverse group. We are now in the process of transitioning the TAO ITS implementation to IITSAT to take advantage of IITSAT's ease of defining and using different instructional methods (IMs). In this new version of TAO ITS, the student is asked a few background questions to assess his level of expertise. If his expertise is low the instructional method is highly structured. He is presented the specific principles and their descriptions in a prescribed order and shown examples of previously recorded simulated scenarios showing the TAO's correct actions which illustrate the principles. The IITSAT version of the TAO ITS then first gives this type of student relatively easy scenarios to practice with. After he has shown that his mastery has reached an intermediate level, then he transitions into the scenario-debrief instructional method and more difficult scenarios. This contrasts with students who are initially (and continue to be) assessed at the intermediate or expert level. These students have more freedom to choose scenarios and are not presented with instruction before scenarios, only debriefed and remediated about their mistakes after the scenario is complete.

The transition of the TAO ITS encountered few problems and took relatively little time partly because it was one of the example applications on which IITSAT's design was based and it contributed its FSM code to IITSAT. Most importantly, the TAO ITS simulator already created a log file of significant events for evaluation purposes in the correct format. Thus, it could be largely used, except has described below, as-is. The major difficulty with the IITSAT version of TAO ITS was based on the fact that for novice TAO students, a hinting mechanism was desired and IITSAT, while allowing the ITS author to specify when hinting would be appropriate, offers no capability to actually provide hints. Thus the hinting mechanism had to be built into the simulator. Additionally, IITSAT communicates the need for hints and other routine information though a specific file format that the simulator had to be altered to read. IITSAT's Instructor Interface had to be upgraded to reflect the capabilities that already existed in the TAO ITS's Instructor Interface Tool (IIT) which included managing students, reviewing their progress, and replaying any of their scenario performances.

GENERAL LESSONS LEARNED

After having applied IITSAT to several specific ITS projects in different domains, there are a large number of general lessons that we have learned relating to using a general ITS authoring tool when creating ITSs. IITSAT's scenario/simulation-based ITS paradigm is good for tactical decision-making (and many other) domains. An ITS authoring tool can save the majority of the software development effort, if the desired ITS is based on the same instructional paradigm on which the authoring tool is based. This will be especially true if there is some flexibility in the desired ITS functionality. However keep in mind two important factors. There will tend to be a high learning curve to use the advanced features/flexibility of the authoring tool. And, some domain specific software will probably need to be written, unless the authoring tool was developed specifically for your domain (i.e. a surface warfare tactical decision-making ITS authoring tool or a mechanized infantry tactical decision-making authoring tool). The most domain specific software in a simulation-based ITS tends to be the simulation and scenario editor. Additionally, some domain specific code will probably need to be written for action/decision correctness evaluation.

Finite state machines (FSMs) are often a good basis for evaluation in dynamic free-play simulated scenarios. They do have some limits. Often they will need to have domain specific primitives written for them, so it is important that the authoring tool allow for this ability to extend itself. Be prepared to write at least some scenario-specific FSMs for each scenario. Furthermore, FSMs will not be able to handle every type of evaluation requiring, again, domain specific code. Especially in more static scenarios (where either the problem doesn't change (i.e. develop a tactical plan, but don't execute it) or student actions do not greatly affect the outcome), comparisons to correct and likely incorrect sets of decisions/actions annotated with appropriate rationale and principles are very helpful in determining correctness. Again, the most detailed part of the comparison will probably be based on primitives using domain-specific code.

Using software not intended for training as the simulator/scenario-player can be very difficult, including just getting it and existing interfaces running. For example, FBCB2 was developed to be a C4I system running in actual vehicles, each equipped with a GPS and connected to a radio network. It was very difficult to get it running on a desktop, without a GPS or radio network connection and to drive the vehicle positions from simulated data. Furthermore it used a highly specific compression scheme (to make the most of the limited bandwidth of the radio network). Similarly using a commercial game, even one that at been interfaced to FBCB2 already, as a basis for a training system had several shortcomings. Foremost was the players access to unrealistic information and the lack of realistic, intelligent behaviors in both friendly and enemy vehicles and units.

Using a file interface between the simulation and ITS had several advantages. Foremost, such interfaces are easier to develop and debug, especially if the interface is a human readable text file. It requires the least modification to an existing simulation, since the simulation only needs to have code to output events to a file added to it. Of course DIS and HLA are potential interface methods, if the simulation already supports them. However, the kind of information the ITS needs may be more detailed that that provided through the HLA or DIS interfaces, which are really intended to provide the data needed to coordinate distributed simulations. This information tends to be just the behavior of the modeled platform that is observable to the external world, such as movement and the use of weapons and sensors. However, the ITS might need to know decisions and actions that the student is taking that are purely internal to his platform. Examples are noting which sectors a tank commander is scanning, plan overlays created in FBCB2, reporting a TAO is supposed to perform during tactical situations, and a team leader correcting mistakes of his subordinates.

An ITS authoring tool may provide more flexibility when the target users or needed capabilities change. That is, these changes may take substantially less development time if the tool provides those capabilities. However an authoring tool may provide less flexibility to implement a new capability if these new capabilities are not present in the authoring tool or at least allowed for. Therefore, ITS authoring tools need an interface to a general purpose language, like C++, and ways to incorporate calculated results back into the authored ITS. Similarly, it is helpful if the authoring tool is under continued development and if its functionality and capabilities can be adjusted and improved for specific uses. Because of the rapid development capabilities, authoring tools are very real helpful for rapid prototyping. Paradoxically, a scenario-based ITS, developed from an operational perspective can be difficult to demonstrate briefly. Different decisions in setting the parameters in the authoring tool would be made if creating a demonstration versus an operational system. Consequently, a "Demo-Mode" would be good addition to an authoring tool that would allow parallel specification of a different set of parameters for use only in demonstrations. These would include the use of smaller, simpler, and fewer scenarios. A more flexible and intelligent use of what scenarios exist when they are few in number would also be helpful. In general, a simulation-based ITS usually demands many scenarios. You will always want more scenarios than you have. Therefore, you should allocate more scenario development time and more scenarios than you think you will actually need.

FUTURE WORK

IITSAT development is continuing. The next version will be completed late in the summer of 2001. Currently an HLA interface is being developed for IITSAT and the student interface is being revamped to make it more intuitive. As more applications are being developed with IITSAT extra features required for them that are generally useful are added back into IITSAT. SHAI has been awarded an additional ITS contract to investigate ways to make ITSs even more adaptive to individual student differences. IITSAT is being used as a basis for this work. Additional adaptive features are being added to it to allow their usefulness to be tested. TAO ITS and the FBCB2/Tactical Decision-Making ITS will be used for this testing.

REFERENCES

Klein, Gary and Zsambok, Caroline E., A. eds. (1997). *Naturalistic Decision Making*. Mahwah, New Jersey: Lawrence Erlbaum Associates, Publishers.

Lussier, James W., Ph.D. (2000). *Coaching Techniques For Adaptive Thinking*, I/ITSEC 2000 Proceedings.

Stottler, Richard H., and Parekh, Sujay S. (November, 1996). *AI Techniques for Reusable Tactics Expert Systems.* Stottler Henke and Associates, Inc., 107-Tactics FR.

Stottler, R. H., and Vinkavich, M. (2000). *Tactical Action Officer Intelligent Tutoring System (TAO ITS)*. I/ITSEC 2000 Proceedings.

Stottler, R. H., and Harmon, N. (2001). *Transitioning an ITS Developed for Schoolhouse Use to the Fleet: TAO ITS, A Case Study*. I/ITSEC 2001 Proceedings.