Rehearsing Naval Tactical Situations Using Simulated Teammates and an Automated Tutor

Emilio Remolina, Sowmya Ramachandran, Richard Stottler, and Alex Davis

Abstract—This paper describes a deployed simulation-based Intelligent Tutoring System (ITS) for training of Tactical Action Officers (TAOs). The TAO on board a Navy ship is responsible for the operation of the entire watch team manning the ship's command center. The ITS goal is to train the TAO in "command by negation," in which watchstanders perform their duties autonomously, while the TAO supervises, intervening in order to correct mistakes and rectify omissions. The ITS uses artificial intelligence (AI) techniques to provide Automated Role Players (ARPs) representing the watchstanders in the ship, and to provide a Natural Language interface to communicate with these automated teammates. An adaptive coaching strategy is used to provide coaching and feedback during an exercise. The paper presents a discussion of the ITS instructional design, its architecture, and the AI techniques it employs.

Index Terms—Artificial intelligence applications, computer-assisted instruction, intelligent tutoring systems.

1 Introduction

The Tactical Action Officer (TAO) on board a US Navy Cruiser, Destroyer, or Frigate is responsible for the operation of the entire watch team manning the ship's command center. Responsibilities include tactical decision making, console operation, communications, and oversight of a variety of watchstander responsibilities in air, surface, and subsurface warfare areas. This paper describes PORTS TAO-ITS, a deployed Intelligent Tutoring System (ITS) [1], [2] for the instruction of TAOs in training at the Surface Warfare Officers School (SWOS) in Newport, Rhode Island [3].

PORTS TAO-ITS uses a learn-by-doing strategy whereby the TAO is presented with a computer-simulated tactical situation, in which he should act as if aboard an actual ship. The ITS uses a high-fidelity simulation of the Aegis system consoles, based on Northrop Grumman PC-based Open-architecture Reconfigurable Training System (PORTS). artificial intelligence techniques are employed to model the behavior of automated crew members who autonomously react to the tactical situation and interact among them and with the TAO. The TAO uses a natural language interface to communicate with the simulated teammates. During an exercise, the ITS provides real-time coaching and feedback which are sensitive to changes in the student's mastery of a wide variety of continually assessed principles.

Before the use of PORTS TAO-ITS, an instructor was needed for every two students. The instructor played the role of other teammates and provided coaching and after action review. The logistics of this training setup provided limited training opportunities to the students. With the advent of the ITS, one instructor is needed in a classroom of 42 students. The ITS teaches material that has the least ambiguity and controversies. These latter types of situations are discussed with instructors and other students and rehearsed using a traditional fully manned simulation.

In the rest of the paper we provide details about the PORTS TAO-ITS functionality (Section 2), its instructional design (Section 3), its architecture (Section 4), and the artificial intelligence techniques it employs (Section 5).

2 PORTS TAO-ITS Functional Description

Most of the TAO's work consists of deciding whether a track (aircraft, boat, submarine) is a friend or a threat, and processing such track accordingly. This requires the TAO to monitor the tracks behavior and to gather additional track information provided by existent human intelligence, electronic sensory information, or by visual identification of the track. In this last case, for example, the TAO could send a helo to identify a far away boat, or have the bridge identify a boat inside visual range. The TAO gathers information, analyzes it, and ensures that the correct decisions are made and actions are taken based on the tactical situation. He does his work by issuing verbal orders and querying the members of the combat information center (CIC). Watchstanders (teammates) autonomously decide which tasks to execute given a tactical situation. They, however, are expected to announce their intentions before executing their tasks. When these are correct, the TAO must merely acknowledge the watchstander's decision. However, if these decisions are incorrect or omitted, the TAO must negate the incorrect decision or proactively initiate the omitted actions [4].

From a functional perspective, PORTS TAO-ITS consists of two major subsystems: 1) PORTS, a high-fidelity simulation of the Aegis system TAOs will use aboard a ship, and 2) the ITS component, which complements the simulation by providing the simulated teammates and the automated tutor that help TAOs learn their jobs by rehearsing tactical situations.

• *The authors are with Stottler Henke Associates, 951 Mariner's Island Blvd., Suite 360, San Mateo, CA 94404.*
E-mail: {remolina, sowmya, stottler, davis}@stottlerhenke.com.

Manuscript received 26 Dec. 2008; revised 23 Mar. 2009; accepted 12 May 2009; published online 15 May 2009.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLTSI-2008-12-0123.
Digital Object Identifier no. 10.1109/TLT.2009.24.

Fig. 1. PORTS TAO-ITS student interface.

Fig. 1 shows the PORTS simulated TAO console and the ITS GUI. Bars (green bars in the figure) are used to display coach feedback and hinting. A control panel is used to display TAO utterances for his approval and to control the scenario execution. The PORTS console includes panels for Variable Action Buttons (VABs), display selection (map control keys), radio control, tactical situation map (a scaled version of the large screen display), and automatic status boards that, among other things, display information on the hooked track (right topmost corner in the figure). The mouse is used to push buttons and select tracks. These displays are driven by a computer simulation of the ownship's sensors and weapons, external platforms, and the environment.

The ITS provides textual hints and feedback using the colored boxes shown in Fig. 1. Each box is associated with a particular principle and expected action the TAO should apply. The color of the box reflects the urgency of the TAO intervention. Most of the TAO actions require verbal communications with simulated teammates. For this, the TAO pushes a pedal and talks. The recognized utterance is shown to the TAO in the control panel. The TAO must click a submit button for the utterance to be sent to other teammates. This extra step of showing the recognized sentence is needed because the system does not recognize all the utterances the student says or because the system could fail to recognize a valid utterance.

2.1 A PORTS-TAO ITS Session

On logging to the ITS, the TAO will execute an exercise that is selected either by an instructor according to a course curriculum or by the ITS itself. In this last case, the ITS suggests exercises that teach principles the student has not mastered. The student is then presented with the exercise's prebrief describing the initial tactical situation. The prebrief is customized for the student by highlighting the principles taught by the exercise that are new to the student or that the student has not mastered. These principles define the aspects of the exercise to which the student should pay the most attention.

When the exercise starts, the PORTS simulator consoles are shown, and the student is free to use the consoles and interact with the simulated teammates at will (Fig. 1). Each exercise has an associated PORTS simulation script that will control the occurrence and behavior of tracks (airplanes, boats, submarines, missiles) that the TAO and simulated teammates might react to (most tracks might be distracters). The behavior of TAO force assets (e.g., the ship, a helo, a P3) is controlled by the simulated teammates as a result of interactions with the TAO. For example, the TAO might decide to send a helo to investigate a track. The ITS will generate the simulation commands that will direct the helo toward the track. Once the helo reaches the track, the simulated teammates inform the TAO of any visual information associated with the track. This information might in turn trigger other behaviors from the teammates, or prompt the TAO to make some actions (e.g., move the helo away from a possibly hostile track). As discussed in Section 3, simulated teammates might make intentional mistakes to force the TAO to give corrective commands.

During the exercise the ITS provides hints when a TAO action is expected and the TAO's mastery of the action's underlying principle is low (do not provide a hint for something the TAO knows how to do). Hinting is faded as the student gains mastery of the domain principles. There are four levels of hints: an alert, a general hint, a specific hint, and a prompt. The alert, the first to appear, is visually similar to other hints but has no content; it consists of a blue bar that appears on the screen. This is the instructional equivalent of a tap on the shoulder, indicating to the student that an action is expected without giving any further information. The remaining hints contain text that gradually becomes more specific as they progress. The general hint tends to indicate some information about the situation that the TAO might not have noticed, to which he is expected to react (e.g., Is your helo available for ASW use?). The specific hint gives the gist of the type of action that is expected (e.g., Use your helo to investigate radar riser.). Finally, the prompt provides the precise content of the action (e.g., Say, ASWE TAO vector helo to investigate radar riser.). The ITS assesses the student differently depending on how many hints are displayed before the correct action is taken, with the lowest assessment resulting from no action at all [4].

On ending the exercise the TAO is given a debrief, a chronological HTML record of principles passed and failed in the exercise, with links to explanations of each principle as well as relevant portions of the detailed exercise transcript. This detailed transcript includes the TAO actions, teammate's interventions, and the significant tactical events that occurred during the exercise (e.g., an incoming missile was detected). The debrief also summarizes any progress the student made. For this, the ITS compares the student model before and after the exercise to identify those new things the student did well or those principles whose proficiency notably decreased.

3 PORTS-TAO ITS Instructional Design

PORT TAO-ITS is designed to support classroom instruction guided by a curriculum. The ITS has two primary modes: classroom and homework. The *classroom mode* is

instructor-led with the entire group of students running the same exercise chosen by the instructor. Here, the instructor introduces the students to exercises that assess principles they have not seen before and, as the exercise goes on, may monitor the progress of the students (and their successes and failures) from a display on the instructor station.

Homework mode allows students to practice on their own, with the ITS selecting exercises containing principles that the student has already encountered in classroom mode and on which the student needs to improve. This mode does not introduce new content, reserving that for the classroom environment where the instructor is available to explain the essence of new principles and expected TAO behavior. The ITS identifies the weakest of those principles already covered by the curriculum (all those principles with comparable lowest proficiency) and selects an exercise that teaches the largest set of those weakest principles. This strategy maximizes the utility of the training scenario by giving the student the opportunity to increase his proficiency in as many principles as possible, while not overwhelming him with the applicability of too many unfamiliar principles. As the student executes scenarios intended to increase proficiency in some principles, the continuous performance evaluation might reveal further deficiencies in other principles, changing the set of weakest principles and thus the focus of the adaptive training. Variety in the training exercises is achieved by creating different simulation scenarios, varying the numbers of tracks or the types of mistakes made by ARPs.

This design was specified by the participating subject matter expert trainers. The rationale was that trainers wanted a greater level of control over the introduction of new concepts.

PORTS TAO-ITS instructional strategy is what TAOs and their trainers call "command by negation." A TAO expects watchstanders to do their jobs correctly, concerning himself with maintaining awareness of those activities and of the tactical picture, stepping in for major decisions such as the engagement of an aircraft. When a watchstander takes an inappropriate action, either by making a mistake or for lack of information not normally available at that watchstation, the TAO negates it and supplies a new order. This also applies when a watchstander fails to act, and the TAO "negates" that inaction. Cases include a watchstander failing to make a routine report on the status of a friendly air asset, or incorrectly announcing intent to query an aircraft when it is within its territorial airspace [4].

The ITS is designed to require this "negation" of the student more and more often, and in response to more nuanced mistakes, over the course of exercises. To this end, we partitioned instructional content into three levels, which were labeled simply by numbers (1)-(3). The three levels apply to principles, Automated Role Players (ARPs) behaviors, and exercises. A single principle may have three levels, in applying to three different kinds of situations as illustrated next.

For example, the principle "query range" consists of knowing the correct range to issue a query to an aircraft. (Whether a query is appropriate for that aircraft in the first place is another principle.) To test this principle at level 1, an ARP announces intent to query at the correct range, and the TAO simply must acknowledge that announcement in order to pass. No negation is necessary. At level 2, the ARP may remain silent when the aircraft reaches the correct range, and the TAO must notice this omission and request the query. At level 3, the ARP may intend to query at the wrong range, which the TAO must notice and correct. The ITS maintains a separate assessment of student's mastery for each principle level so that the student must learn how to act correctly in each type of situation.

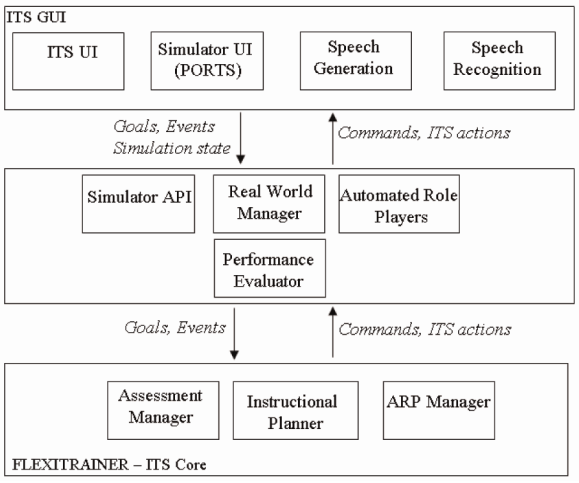

Fig. 2. PORTS TAO-ITS architecture.

Some principles do not involve ARP's actions at all, but rather require the TAO to maintain situational awareness of the tactical picture, performing console actions to select tracks and varying display ranges appropriately to cover the area referred by the current tactical conversation being carried out among ARPs.

Predictability in exercises was an important design requirement in order to make it easy for instructors to monitor students. With this in mind various events controlling the tactical situation are specified when authoring the exercise and are not dependent on the student model. Given a scenario, coaching is the only dimension of adaptation to student expertise.

4 PORTS-TAO ITS Architecture

Fig. 2 shows a simplified diagram of the PORTS TAO-ITS architecture. The figure does not include the different authoring tools, content elements (e.g., principles, exercises), and external applications (e.g., LMS, instructor console) that are part of the deployed system. Next, we discuss the three loosely coupled layers in the architecture.

The ITS GUI layer includes those components the student interacts with during an exercise. These components are

- *ITS UI*: This is the user interface that enables the student to start an exercise and review the debrief after the exercise is complete.
- *Simulator UI*: This is the student GUI used when executing an exercise. It consists of the underlying PORTS interface and an UI that allows the ITS to

provide hints and real-time feedback to the student (see Fig. 1).

- *Speech Generation*: This is the module to convert ARP's textual utterances to speech. The ITS uses ATT Natural Voices text-to-speech software in its implementation.
- *Speech Recognition*: This is the module to convert student speech to an internal representation (events). The ITS uses SRI EduSpeak software in its implementation.

The ITS middle layer encapsulates most of the domain-dependent knowledge required for the simulated teammates' behaviors and the student evaluations. The main components are

- *Simulator API*: This component is a wrapper around the disparate PORTS APIs to simplify interaction between the ITS and PORTS. It allows for the receipt of simulation events (e.g., "weapon launched") as well as querying the simulation's current state. It also allows the ITS to send commands to PORTS to modify the state of the simulation dynamically (e.g., control a friendly helo to carry out TAO orders).
- *Real-World Manager*: This component maintains a world representation suitable for reasoning about the tactical situation. This representation enhances the simulation data with domain concepts that are not represented by the simulator (e.g., air corridor, track groups).
- *Automated Role Players (ARPs)*: This component is responsible for simulating the behaviors of the various teammates in an exercise. The ARPs are capable of monitoring the state of the simulation and generating utterances as well as actions in the simulation.
- *Performance Evaluator*: This component is responsible for monitoring the student's performance. It runs *evaluation machines*, each in charge of evaluating the student in a specific well-defined situation that requires the TAO to apply a principle (a particular set of skills and knowledge). Evaluation machines process student utterances, actions, and the current simulation state and in turn generate abstract events that signal when the student should be applying a principle and whether the student failed or succeeded in doing so. The coach and assessment agents will interpret these events, effectively decoupling evaluation, coaching, and performance assessment.

The ITS core layer is based on Flexitrainer [5], an extensible, agent-based ITS framework that handles instructional planning, updates the student model, manages all interactions with the student, and coordinates the various components of the system. Its main components are

- *Instructional Planner*: At the heart of the FlexiTrainer engine, there is a collection of "agents" responsible for carrying out instructional actions [6]. The Instructional Planner is responsible for selecting which agents should carry out what actions in order to achieve the goal posted to it. Adding functionality to the FlexiTrainer engine is done by adding new agents or modifying the actions/behavior of existing ones. The ITS includes a coach agent that monitors the student's performance and generates appropriate feedback, hinting, and coaching. There is also a debriefer agent that generates the after action review, and an exercise picker agent that implements the algorithms to suggest exercises when in homework mode (Section 3).
- *Assessment Manager*: This component is responsible for all functionality relating to cumulative assessment diagnosis. By cumulative assessments, we mean assessments that are aggregated over the long term (multiple scenario performances and observations). It makes decisions about the student's proficiency on tasks, skills, and principles, and performs differential diagnosis (e.g., it assess that the student can perform a skill under certain conditions but not in others).
- *ARP Manager*: It provides the interface between the ARPs and instructional agents embedded in FlexiTrainer, allowing FlexiTrainer to send commands to the ARPs. For example, the coach agent will instruct from time to time an ARP to make a mistake for instructional purposes. The ARP in turn will generate a corresponding ARP event indicating that it has made a mistake (and hence, a TAO corrective action is now expected).

In this architecture, components interact with each other mostly through events. A typical interaction among components will be as follows. Suppose the simulated AIR teammate requests authorization to query a track, and that the TAO is expected to acknowledge the communication. The AIR sends a command to the text-to-speech component, and once the command is executed, it publishes an event capturing the intent of its utterance, something like $\{ARPRequestForTrackActiontrackId = 80001, action = query, source = AIR, destination = TAO\}$. This event is received among others by the evaluation machines associated with the principles evaluated by the exercise. One of such evaluation machines will deduce that the TAO needs to apply the principle "query-range-1" which requires the TAO to acknowledge the query by saying "TAO aye." The evaluation machine then publishes an event signaling that principle evaluation is active, an event like $\{EvalPrinciple, principle = query\text{-}range\text{-}1, trackId = 80001\}$. This event is received by the coach agent, who in turn starts a behavior to provide hints for the principle "query-range-1" (if the student mastery of the query-range-1 principle is low). Suppose the TAO says "TAO aye" after the first hint is shown. This utterance will be translated into some event that the evaluation machine and the AIR will receive. The AIR will proceed to do the query. The evaluation machine will detect a successful application of the principle and publish an event signaling success. The coach agent will receive this event, relate it to the current coaching, and provide a positive real-time feedback. Conversely, if the TAO does not take any action, then a principle failure will be signaled. The "query-range-1" has associated time-outs indicating when a hint should be given and the maximum time allowed for the TAO response. If this maximum time passes without the TAO action, the evaluation machine

publishes a failure event. The coach will receive this event and provide a negative real-time feedback to the student.

5 Artificial Intelligence Techniques

The main artificial intelligence techniques employed by PORTS-TAO ITS are the use of explicit domain ontologies, natural language processing, and modeling of human behaviors. Next, we discuss the use of these techniques.

5.1 Knowledge Representation

The ITS uses an explicit domain ontology representing the domain concepts and relationships among these concepts [7]. Although many of the simulator concepts have a counterpart on this ontology, this ontology is mainly about domain concepts (e.g., tracks, track group, tasks, events) and ITS concepts (e.g., principles, exercises, hints). Tracks and track groups represent simulation objects and the other concepts are needed to support the instructional agents. The state of the simulation is represented by the physical and geographical information about tracks (e.g., the location and speed of an airplane) and the collection of actions the ARPs and the TAO have taken with respect to such tracks (e.g., whether a track has been queried). Explicit relationships among tracks are also maintained, for instance, to know which tracks belong to a formation, or which missiles have been launched to counter an enemy airplane.

Rules are used to propagate the properties of tracks in a group from those of individual units. For example, a set of boats will be declared to be in a group if they are close to each other and heading in the same direction at about the same speed. The leader of the group will be the boat in a group that is closer to the ownship. The TAO and the ARPs will process a group of tracks by referring to the group leader and any action on the group leader is understood to apply to the whole group. A set of "ontology rules" capture the fact that if a group leader is declared friendly, then all other group tracks are friendly too. These rules simplify the definition of ARP's behaviors by eliminating the need to explicitly set the properties of each unit in a group when a property for the group leader changes.

Finally, rules of engagement are captured within the ARP behaviors. We illustrate this in the next section when an ARP is asked to query a track.

5.2 Natural Language Processing

The ITS uses the commercially available EduSpeak speech recognition software to transform speech into text. EduSpeak is used in a command-and-control mode, where it only recognizes utterances included in a given grammar. This grammar increases the recognition rate and was decided appropriate for this application [4].

Although the use of a grammar helps with the speech recognition problem, the ITS grammar allows indexicals that require some contextual information for a sentence to be interpreted. The ITS does not use a central component that provides an internal representation for every possible utterance (a well-known hard problem), but rather the ITS uses a distributed approach where the final interpretation of an utterance is left to the simulated teammates and evaluation machines processing such utterances. Simulated teammates and evaluation machines use some context from ongoing dialogs (represented internally by ARP's behaviors) and the tactical situation (as represented by the track information) to understand a sentence. At times a sentence just might be ignored. Next, we illustrate the use of this contextual information.

The EduSpeak's text is transformed into a set of events representing the intent of the utterance. Simulated teammates and evaluation machines will receive these events like any other events in the system. Most TAO utterances are transformed into events that have all the needed information for an ARP to decide how to react. For example, if the TAO says "AIR TAO query track 80001," an event like $\{RequestTrackActionaction = ``query''trackId = 80001, source = TAOdestination = AIR\}$ will be generated and the AIR teammate will be able to check if the query should be performed (as specified by the rules of engagement), and start the behavior that will perform the query if required. The similar sentence "AIR TAO, negative, query track" will not specify the track and the AIR might not know what the sentence is about. If the AIR has recently made the mistake of "warning track 80001 before querying it," then the AIR will be expecting a query order and will interpret the TAO sentence as "query track 80001." The AIR will then acknowledge the TAO and start the query procedure. In any other context, the AIR will just remain silent. The details of how ARPs maintain such context information are explained in the next section.

5.3 Automated Role Players

The ITS includes 30 simulated teammates, performing 370 behaviors, and evaluates the TAO in 160 principles across the air, surface, and subsurface warfare areas. The behaviors of simulated teammates are modeled using Behavior Transition Networks (BTNs) [8], [9]. An BTN is a kind of state machine where states are behaviors and state transitions represent the conditions under which an agent will stop executing one behavior and start executing another behavior. Basic behaviors (those that cannot be described in terms of other behaviors) are called actions. Unlike state machines, BTNs provide the following constructs proper of programming languages:

1. they are hierarchical,
2. have variables (local memory),
3. have access to blackboards (shared memory),
4. are polymorphic—the same behavior is executed differently by each teammate, and
5. can execute arbitrary perceptual or action-oriented code (e.g., query a database, interact with the simulator).

The ITS uses the SimBionic tool to author and execute BTNs [8]. As illustrated in Fig. 3, SimBionic allows scenario authors to create behaviors by drawing a flowchart-like graphical representation. Specifically, actions are represented as rectangles, behaviors as boldfaced rectangles, conditions as ovals, and connectors as lines. Scenario authors can attach as many variable assignments, complex expressions, and explanatory comments as they like to any of these elements.

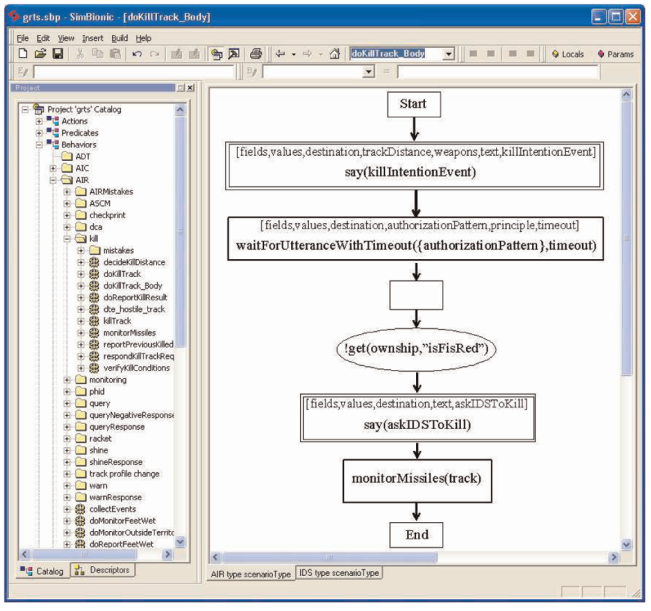

Fig. 3. Representing simulated teammate behaviors using BTNs.

When modeling using BTNs, authors must decide on the basic building blocks (actions, predicates, and key behaviors shared by simulated entities) and use these building blocks to define specific behaviors for the simulated teammates in the scenario. The actual modeling process most often follows a top-down design where high-level behaviors are successively refined. Next, we present some actions, predicates, and behaviors used to model simulated teammates in the PORT TAO-ITS application.

Simulated teammates actions represent the primitive things an agent can do. Some actions are domain independent, like generating an event, or the action of sending a request to the text-to-speech engine. Some actions are domain dependent, like matching a submarine datum against a database of datum. Simulated teammates' predicates are used to check conditions or to access data. Domain-independent predicates include checking whether a new event has been received or checking whether a command has been executed. Domain-dependent predicates include calculating the CPA (Closest Point of Approach) of a given track and most other geometric calculations proper of the domain.

Each ARP has a possibly large number of associated behaviors, each representing how the ARP should react to a particular situation. Fig. 3 shows a high-level description of how the simulated AIR teammate might execute a kill track procedure. In this procedure, the AIR will first announce its intent to kill, wait for the TAO authorization, wait for the ship's FIS to be green (a switch the TAO must turn before engagements), order the IDS teammate to launch the missile, and then monitor that the missile hits its target. The definition of this behavior uses domain-independent "low-level" behaviors, like the ones to say something, and the ones to wait for an authorization (wait for an event of a given type satisfying some constraints). The definition also uses domain-dependent behaviors like monitoring that a missile hits a target. The sequential logic of the behavior is defined by the arrows connecting the boxes in Fig. 3. As expected, not all ARP behaviors will follow a linear logic illustrated in Fig. 3.

Fig. 3 also illustrates some of the expressiveness of the BTNs language. Notice, in particular, how the "wait for the ship's FIS to be green" is simply represented by a behavior transition conditioned in the predicate "get(ownship,"isFisRed")" to be false (oval in Fig. 3). If switching the FIS to green counts as an kill authorization, then the arrow leaving the "waitForUtterance" box will be directly connected to the oval labeled by "get(ownship,"isFisRed")." This transformation will indeed represent the fact that the "wait for authorization" behavior should be suspended as soon as the FIS is green. Behavior representation constructs like this have been proposed in different AI agent literature, especially in the area of mobile robotics [10].

Behaviors associated with simulated teammates are triggered by the state of the simulation (e.g., a track is approaching the ship), or by events generated by other simulated teammates (e.g., the AIR teammate requests all other teammates to checkprint a given track). This way, the behavior of the team is controlled by changing the environment conditions where the team interacts. This approach circumvents the problem of defining tailored-simulated teammates and changing the simulation to support a particular scenario. The logic of each ARP behavior might itself be simple, but a complex system behavior emerges from the complexity of the environment in which the ARPs and the trainee are operating. This is akin to the "reactive agents" approach proposed in the AI literature [11].

Information derived from the execution of a behavior is stored by either updating objects included in the world representation (e.g., mark a track as checkprinted, declare a track hostile) or by making assertions in a *blackboard* used to represent the current tactical context. For example, this blackboard includes the current threat (if any), information that is used to disambiguate TAO utterances using indexicals or qualifications that refer to such a threat (e.g., "Surface TAO query incoming boat"). Each ARP has specific behaviors that contribute to add or retract information from this blackboard and therefore maintain the current context current.

Dialogs with other teammates or humans (there is not a difference) are represented by BTNs showing the logic of the agent's interventions during the dialog (Fig. 3). Each agent uses its blackboard to store relevant information that may affect the dialog execution. The agent interventions are usually conditioned on information contained in this blackboard (e.g., avoid asking for information already provided in the dialog). Dialogs between the trainee and the simulated teammates may take different courses depending mainly on the information provided by the trainee. Many dialogs follow the same pattern of interactions or dialog rules. The system uses a library of dialog macros (implemented by BTNs) that abstract these common interactions and help simplify the authoring of dialogs.

As discussed in Section 3, the ITS is designed to train the TAO by "command by negation," where the TAO has to

know the job of the teammates and acknowledge correct actions or correct incorrect actions. Accordingly, ARP behaviors exist in order to create situations where principles can be evaluated for all levels. Level 1 is intended to help the TAO to learn to maintain awareness, and to habituate to the normal responsibilities of the watchstanders in a variety of situations. At this level, ARPs perform perfectly and the student witnesses no actions that need negating, so usually the TAO is simply expected to verbally acknowledge the action.

Level 2 requires the TAO to apply this awareness, by noticing when the usual procedures are not followed. At this level, ARPs tend to make mistakes of omission, usually by remaining silent when an utterance is appropriate. This requires the TAO to notice the omission and take action, most often by requesting the ARP to act. When the TAO does not act, ARPs might correct the situation themselves, or another ARP (the Captain) might intervene to correct the situation.

Level 3 involves the ARPs tending to make mistakes of commission, where actions are taken incorrectly. Here, the TAO is expected to notice the mistake and request a correction of it. In the middle of an exercise, with many concurrent threads of activity with respect to a variety of simultaneous tracks, these ARP mistakes can be harder to notice and correct than when an ARP omitted an action altogether.

ARPs do not commit mistakes at every possible opportunity, but rather do their jobs correctly until a mistake is designated to happen. This is under control of the exercise author, who decides what mistakes will happen in an exercise, during what time range, and with what frequency. Thus, the body of content on which the student is evaluated in a particular exercise is tightly controlled, but there is randomness to the mistakes from the student's perspective to prevent the student from "learning" the exercise rather than the principles. Thus, two exercises may have identical sets of events within the tactical picture, but depending on the ARP behaviors, may evaluate entirely different principles.

Finally, evaluation machines are also implemented using BTNs. In this case, the purpose of the evaluation machine behavior is to signal that the TAO should be applying a principle and then decide whether the TAO executed the correct action. The best way to employ BTNs to monitor real-time mission execution is to have a large number operating in parallel, where each looks at the situation and student's actions from the perspective of how they handle specific types of situations or apply specific types of principles. Since most of the TAO principles require the supervision of the other watchstanders, the implementation of the evaluation machines is simplified by having the ARPs explicitly signal when they do a correct action, omit an action, or take an incorrect action. This way, the evaluation machines do not have to continually monitor the ARPs to decide how the TAO should act.

6 Related Work

The PORT TAO-ITS uses a scenario-based learn-by-doing training approach as proposed in [12]. This approach is akin to the Constraint-Based Tutors applied in intractable domains, in which knowledge cannot be fully articulated, and it is considered impossible or impractical to build a computational system which can perform at the level of the human expert [2]. Instead of building a general expert model to interpret student actions [1], [13], a collection of situated experts (evaluation machines) is designed to handle a specific type of situation that might arise during a training scenario. Each evaluation machine is activated by its own set of preconditions. These are much easier to develop in a case-specific way than in the general case. By parameterizing these evaluation machines, it is possible to use them in a variety of tactical situations.

The idea of training with simulated teammates is not new. For example, Johnson et al. [14] discuss the key capabilities and technical issues of using animated pedagogical agents and their potential for team training among others. In [15], an architecture and modeling language is proposed to define such agents so that synthetic teammates exhibit flexible human-like behavior and are able to support face-to-face spoken conversations in virtual worlds.

The need for simulated teammates to train tactical situations under stress conditions, like the ones described in this paper, has long been recognized by the military, either as a cost-effective training alternative (to reduce instructor-student ratio) or as a response to concerns generated by on-the-job experiences. For example, the Tactical Decision Making Under Stress (TADMUS) project, sponsored by the Office of Naval Research (ONR), was spawned by the 1988 USS Vincennes incident, where an Aegis cruiser engaged in a littoral warfare peace-keeping mission shot down an Iranian Airbus. Investigations following the incident suggested that stress may have effects on decision making and that these effects were not well understood [16]. Also, the Synthetic Cognition for Operational Team Training (SCOTT) project proposes to apply advanced human behavioral representation methods to meet a critical team training need in the Navy, specifically to provide deployable training for critical aviation team skills without requiring any additional personnel besides the individual receiving the training [17], [18].

Different approaches and representation languages have been used to model teamwork issues such as team roles, authority, coordination, and negotiated decision making [15], [19], [20], [21], [22]. These approaches share the common idea that agents need to have an explicit representation of tasks, goals, and shared situational state that allow team members to monitor the environment, proactively carry out actions, give or accept orders, and negotiate plan options. These approaches differ in aspects like their underlying teamwork theories (e.g., joint intentions [23] is used by BDI agents [19] and in STEAM [20], sharedPlans in [22]) and the type of out-of-the-box functionality embedded in the representation language and the companion execution algorithms. For example, STEAM [20] provides algorithms that automatically establish commitments to a team goal and to reassign team member responsibilities if necessary. In [17], a rich model of dialogs is linked with the task model both to interpret utterances as well as to decide when the agent should speak and what to say.

In our application, we used BTNs to model the simulated teammate's behaviors. BTNs complement "production rules" used in systems such as Soar [24] and iGEN

[17], by using the hierarchical state machine formalism to facilitate the definition of task sequencing and branching logic. However, BTNs do not provide any special construct to support teamwork. There are, however, some teamwork aspects that simplify the implementation of simulated teammates in our application. The team structure is fixed and the responsibilities of team members are well established. Authority and commitment to team goals are not an issue either. The major issues in our application are 1) to decide which actions should be taken and coordinate with other teammates the action execution, 2) to maintain a situation representation supporting this action selection and execution, and 3) to carry out dialogs with the student and other teammates. As described in Section 5, we have implemented content-independent BTNs that capture some common coordination patterns proper of this domain. These patterns are used to define situation-dependent behaviors where the team coordination aspects are hidden at the highest level of a behavior description (following the hierarchical structure of a behavior will reveal these coordination aspects). Similarly, dialogs are implemented like any other agent behavior (using BTNs), the logic of these behaviors describing the conditions under which the agent should intervene. This approach has sufficed for this application, but we are considering the use of specialized modules as in [17] that might simplify the authoring of dialogs.

7 Conclusion

PORTS TAO-ITS illustrates a simulation-based learn-by-doing tactical decision-making ITS, where the student interacts with simulated teammates using a natural language interface. Behavior Transition Networks (BTNs) facilitate the definition of the simulated teammates' behaviors and evaluation machines that provide the situated context for real-time coaching and after action review. Evaluation machines circumvent the problem of creating an expert model of the TAO behavior or their simulated teammates in order to provide tutoring. The creation of such expert models might be impossible in this domain.

The most risky aspect of the ITS is the use of a natural language interface. The ITS largely bases its assessment of the student's performance on the output of the speech recognition system. The speech recognition system uses a grammar including correct command syntax and vocabulary augmented with likely incorrect replacement words (synonyms such as "kill," "engage," "destroy," etc.). Instructors felt this was appropriated since this system was not being designed to tutor radio communications syntax and skills. Thus, communication vocabulary or syntax mistakes would not be caused by a lack of tactical understanding but were more likely to be the results of momentary lapses. This design decision turned out to be appropriated for beginner and intermediate students but not for expert TAOs who usually want to use a more loose language.

The deployment of PORTS TAO-ITS took 2 years ending in October of 2008. Different versions of the ITS have been used in classroom instruction as part of the TAO training at SWOS. Although there have been informal evaluations showing its potential benefit, it is still early to have some conclusive evaluations of its instructional value. The final acceptance and success of the ITS not only depends on its technical and instructional value, but in the organizational commitment to this unfamiliar technology.

References

[1] *Foundations of Intelligent Tutoring Systems,* M.C. Polson and J.J. Richardson, eds., Lawrence Erlbaum Assoc., 1988.

[2] B. Woolf, *Building Intelligent Interactive Tutors: Student-Centered Strategies for Revolutionizing E-Learning.* Morgan Kaufmann Publishers, 2008.

[3] R. Stottler and S. Panichas, "A New Generation of Tactical Action Officer Intelligent Tutoring System (ITS)," *Proc. Industry/Interservice, Training, Simulation and Education Conf. (I/ITSEC),* 2006.

[4] R. Stottler, A. Davis, S. Panichas, and M. Treadwell, "Designing and Implementing Intelligent Tutoring Instruction for Tactical Action Officers," *Proc. Industry/Interservice, Training, Simulation and Education Conf. (I/ITSEC),* 2007.

[5] S. Ramachandran, E. Remolina, and D. Fu, "Flexitrainer: A Visual Authoring Framework for Case-Based Intelligent Tutoring Systems," *Proc. Seventh Int'l Conf. Intelligent Tutoring,* pp. 848-850, 2004.

[6] M.J. Wooldridge, *An Introduction to MultiAgent Systems,* John Wiley & Sons, 2002.

[7] B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins, "What Are Ontologies, and Why Do We Need Them?" *IEEE Intelligent Systems,* vol. 14, no. 1, pp. 20-26, Jan. 1999.

[8] D. Fu and R. Houlette, "Putting AI in Entertainment: An AI Authoring Tool for Simulations and Games," *IEEE Intelligent Systems,* vol. 17, no. 4, pp. 81-84, July/Aug. 2002.

[9] D. Fu, R. Houlette, and J. Ludwig, "An AI Modeling Tool for Designers and Developers," *Proc. IEEE Aerospace Conf.,* 2007.

[10] *Artificial Intelligence and Mobile Robotics,* D. Kortenkamp, R.P. Bonasso, and R. Murphy, eds. MIT Press, 1998.

[11] R.A. Brooks, "Intelligence without Representation," *Artificial Intelligence,* vol. 47, pp. 139-160, 1991.

[12] R. Stottler and S. Ramachandran, "A Case-Based Reasoning Approach to Intelligent Tutoring Systems (ITS) and ITS Authoring," *Proc. FLAIRS Conf. Special Track on Intelligent Tutoring Systems,* 1999.

[13] J.R. Anderson, "The Expert Module," *Handbook of Intelligent Training Systems,* M. Polson and J. Richardson, eds., pp. 21-53, Lawrence Erlbaum Assoc., 1988.

[14] W.L. Johnson, J.W. Rickel, and J.C. Lester, "Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments," *Int'l J. Artificial Intelligence in Education,* vol. 11, pp. 47-78, 2000.

[15] D. Traum, J.W. Rickel, J. Gratch, and S. Marsella, "Negotiation over Tasks in Hyrid Human-Agent Teams for Simulation-Based Training," *Proc. Int'l Joint Conf. Autonomous Agents and Multiagent Systems,* pp. 44-448, 2003.

[16] *Making Decisions Under Stress: Implications for Individual and Team Training,* J.A. Cannon-Bowers and E. Salas, eds. APA Books, 1998.

[17] J. Scolaro and T. Santarelli, "Cognitive Modeling Teamwork, Taskwork, and Instructional Behavior in Synthetic Teammates," *Proc. 11th Conf. Computer Generated Forces,* 2002.

[18] W. Zachary, T. Santarelli, J. Ryder, J. Stokes, D. Scolaro, D. Lyons, M. Bergondy, and J. Johnston, "Using a Community of Intelligent Synthetic Entities to Support Operational Team Training," *Proc. 10th Conf. Computer Generated Forces,* pp. 215-224, 2001.

[19] A.S. Rao and M.P. Geofgeff, "Modeling Rational Agents within a BDI Architecture," *Proc. Second Int'l Conf. Principles of Knowledge Representation and Reasoning,* pp. 473-484, 1991.

[20] M. Tamble, "Towards Flexible Teamwork," *J. Artificial Intelligence Research,* vol. 7, pp. 83-124, 1997.

[21] J. Yen, J. Yin, T.R. Ioerger, M.S. Miller, D. Xu, and R.A. Volz, "CAST: Collaborative Agents for Simulating Teamwork," *Proc. Int'l Joint Conf. Artificial Intelligence,* pp. 1135-1142, 2001.

[22] B. Grosz and S. Kraus, "Collaborative Plans for Complex Group Action," *Artificial Intelligence,* vol. 86, pp. 269-357, 1996.

[23] P.R. Cohen and H.J. Levesque, "Teamwork," *Nous,* vol. 25, no. 4, pp. 487-512, 1991.

[24] J. Laird, A. Newell, and P. Rosembloom, "Soar: An Architecture for General Intelligence," *Artificial Intelligence,* vol. 33, no. 1, pp. 1-64, 1987.

Emilio Remolina received the PhD degree in computer science from the University of Texas at Austin in 2001. He is an artificial intelligence research scientist at Stottler Henke Associates, Inc. He has been the main designer and developer of different ITS systems: ICT (Intelligent Counter Intelligence in Combating Terrorism Tutor), CITTP (Computerized Individual Trainer for Team Performance), AIS-IFT (helicopter flying trainer), and PORTS TAO-ITS (tactical training of TAOs using the AEGIS system). His research interests include intelligent tutoring systems, planning, simulation, and common sense reasoning.

Sowmya Ramachandran received the PhD degree in computer science from the University of Texas at Austin in 1998. She is a research scientist at Stottler Henke Associates, Inc. Her interests focus on intelligent training and education technology, including intelligent tutoring systems and intelligent synthetic agents for simulations. She has developed ITS for a range of topics, including reading comprehension, high-school algebra, helicopter piloting, and healthcare domains. She has participated in workshops organized by the Learning Federation, a division of the Federation of American Scientists, to lay out a roadmap for critical future research and funding in the area of ITS and virtual patient simulations.

Richard Stottler received a master's degree in computer science from Stanford University. He cofounded Stottler Henke Associates, Inc., an artificial intelligence consulting firm in San Mateo, California, in 1988. He has been the principal investigator on a large number of tactical decision-making intelligent tutoring system projects conducted by Stottler Henke including projects for the US Navy, US Army, US Air Force, and US Marine Corps.

Alex Davis received the MS degree in computer science from the State University of New York at Buffalo. He is an artificial intelligence researcher at Stottler Henke Associates, Inc. He has served as a lead knowledge and software engineer for a variety of projects related to artificial intelligence, including simulations, behavior modeling for automated agents, and intelligent tutoring systems.