

## Virtual Humans for Team Training in Virtual Reality

Jeff Rickel and W. Lewis Johnson

Information Sciences Institute & Computer Science Department  
University of Southern California  
4676 Admiralty Way, Marina del Rey, CA 90292-6695  
rickel@isi.edu, johnson@isi.edu  
<http://www.isi.edu/isd/VET/vet.html>

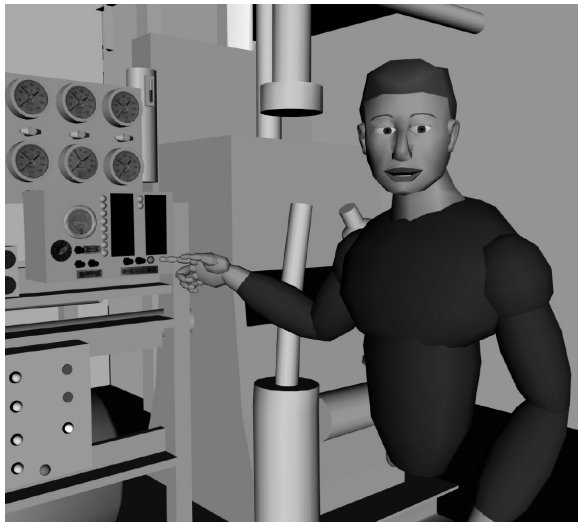


Figure 1: Steve describing an indicator light

### Abstract

This paper describes the use of virtual humans and distributed virtual reality to support team training, where students must learn their individual role in the team as well as how to coordinate their actions with their teammates. Students, instructors, and virtual humans cohabit a 3D, simulated mock-up of their work environment, where they can practice together in realistic situations. The virtual humans can serve as instructors for individual students, and they can substitute for missing team members, allowing students to practice team tasks when some or all human instructors and teammates are unavailable. The paper describes our learning environment, the issues that arise in developing virtual humans for team training, and our design for the virtual humans, which is an extension of our Steve agent previously used for one-on-one tutoring.

### 1 Introduction

Complex tasks often require the coordinated actions of multiple team members. Team tasks are ubiqui-

tous in today's society; for example, teamwork is critical in manufacturing, in an emergency room, and on a battlefield. To perform effectively in a team, each member must master their individual role *and* learn to coordinate their actions with their teammates. There is no substitute for hands-on experience under a wide range of situations, yet such experience is often difficult to acquire; required equipment may be unavailable for training, important training situations may be difficult to re-create, and mistakes in the real world may be expensive or hazardous. In such cases, distributed virtual reality provides a promising alternative to real world training; students, possibly at different locations, cohabit a three-dimensional, interactive, simulated mock-up of their work environment, where they can practice together in realistic situations.

However, the availability of a realistic virtual environment is not sufficient to ensure effective learning. Instructors are needed to demonstrate correct performance, guide students past impasses, and point out errors that students might miss. Yet requiring instructors to continually monitor student activities places a heavy burden on their time, and may severely limit students' training time. In addition, team training requires the availability of all appropriate team members, and may require adversaries as well. Thus, while virtual environments allow students to practice scenarios anywhere and anytime, the need for instructors and a full set of teammates and adversaries can provide a serious training bottleneck.

One solution to this problem is to complement the use of human instructors and teammates with intelligent agents that can take their place when they are unavailable. The intelligent agents cohabit the virtual world with human students and collaborate (or compete) with them on training scenarios. Intelligent agents have already proven valuable in this role as fighter pilots in large battlefield simulations (Hill *et al.* 1997; Jones, Laird, & Nielsen 1998), but such agents have a limited ability to interact with students. Our

work focuses on a different sort of agent: a virtual human that interacts with students through face-to-face collaboration in the virtual world, either as an instructor or a teammate. We call our agent Steve (Soar Training Expert for Virtual Environments).

Our prior work focused on Steve's ability to provide one-on-one tutoring to students for individual tasks (Rickel & Johnson 1997a; 1997b; 1999). Steve has a variety of pedagogical capabilities one would expect of an intelligent tutoring system. For example, he can point out student errors, and he can answer questions such as "What should I do next?" and "Why?". However, because he has an animated body, and cohabits the virtual world with his student, he can provide more human-like assistance than previous disembodied tutors. For example, he can demonstrate actions, use gaze and gestures to direct the student's attention, and guide the student around the virtual world. This makes Steve particularly valuable for teaching tasks that require interaction with the physical world.

This paper describes our extensions to Steve to support team training. Steve agents can play two valuable roles: they can serve as a tutor for an individual human team member, and they can substitute for missing team members, allowing students to practice team tasks without requiring all their human teammates. Steve's prior skills provided a solid foundation for his roles in team training, but several new issues had to be addressed. Each agent must be able to track the actions of multiple other agents and people, understand the role of each team member as well as the interdependencies, and communicate with both human and agent teammates for task coordination. In the remainder of this paper, we describe our learning environment for team training (Section 2), our solutions to these issues (Section 3), and related and future work (Section 4).

## 2 The Learning Environment

Our learning environment is designed to mimic the approach used at the naval training facility in Great Lakes, Illinois, where we observed team training exercises. The team to be trained is presented with a scenario, such as a loss of fuel oil pressure in one of the gas turbine engines that propels the ship. The team must work together, guided by standard procedures, to handle the casualty. At Great Lakes, the team trains on real, operational equipment. Because the equipment is in operation, the trainers have limited ability to simulate ship casualties; for example, they must mark gauges with grease pencils to indicate hypothetical readings. In our learning environment, the team, consisting of any combination of Steve agents and hu-

man students, is immersed in a simulated mock-up of the ship; the simulator creates the scenario conditions. As at Great Lakes, each student is accompanied by an instructor (human or agent) that coaches them on their role.

Each student gets a 3D, immersive view of the virtual world through a head-mounted display (HMD) and interacts with the world via data gloves. Lockheed Martin's Vista Viewer software (Stiles, McCarthy, & Pontecorvo 1995) uses data from a position and orientation sensor on the HMD to update the student's view as she moves around. Additional sensors on the gloves keep track of the student's hands, and Vista sends out messages when the student touches virtual objects. These messages are received and handled by the simulator, which controls the behavior of the virtual world. Our current implementation uses VIVIDS (Munro & Surmon 1997), developed at the USC Behavioral Technology Laboratories, for simulation authoring and execution. Separate audio software broadcasts environmental noises through headphones on the HMD based on the student's proximity to their source in the virtual world. Our current training environment simulates the interior of a ship, complete with gas turbine engines, a variety of consoles, and their surrounding pipes, platforms, stairs and walls. A course author can create a new environment by creating new graphical models, a simulation model, and the audio files for environmental sounds.

Our architecture for creating virtual worlds (Johnson *et al.* 1998) allows any number of humans and agents to cohabit the virtual world. While the behavior of the world is controlled by a single simulator, each person interacts with the world through their own copy of Vista and the audio software, and each agent runs as a separate process. The separate software components communicate by passing messages via a central message dispatcher; our current implementation uses Sun's ToolTalk as the message dispatcher. This distributed architecture is modular and extensible, and it allows the various processes to run on different machines, possibly at different locations. This approach greatly facilitates team training, where arbitrary combinations of people and agents must cohabit the virtual world; our extension to team training would have been more difficult had we originally designed a more monolithic system geared towards a single student and tutor.

Humans and agents communicate through spoken dialogue. An agent speaks to a person (teammate or student) by sending a message to the person's text-to-speech software, which broadcasts the utterance through the person's headphones. Our current im-

plementation uses Entropic’s TrueTalk for speech synthesis. When a person speaks, a microphone on their HMD sends their utterance to speech recognition software, which broadcasts a semantic representation of the utterance to all the agents. The person starts speech recognition prior to each utterance by touching their two index fingers together; Vista detects the signal from the gloves and sends a message to activate that person’s speech recognition software. Our current implementation uses Entropic’s GraphVite for speech recognition. Currently, Vista provides no direct support for human-to-human conversation; if the humans are not located in the same room, they must use telephone or radio to hear one another.

For team training, teammates and instructors must be able to track each other’s activities. Each person sees each other person in the virtual world as a head and two hands. The head is simply a graphical model, so each person can have a distinct appearance, possibly with their own face texture-mapped onto the graphical head. Each Vista tracks the position and orientation of its person’s head and hands via the sensors, and it broadcasts the information to agents and the other Vistas. Each agent appears as a human upper body (as shown in Figure 1). To distinguish different agents, each agent can be configured with its own shirt, hair, eye, and skin color, and its voice can be made distinct by setting its speech rate, base-line pitch, and vocal tract size (these parameters are supported by the TrueTalk software). The agents, of course, do not need audio or visual cues to distinguish other agents and humans; each Vista and speech recognizer indicates in its messages which person it is tracking, and agents send out similar messages about their activities.

### 3 Agent Design

#### 3.1 Architecture

Each Steve agent consists of three main modules: perception, cognition, and motor control (Rickel & Johnson 1999). The perception module monitors messages from other software components, identifies relevant events, and maintains a snapshot of the state of the world. It tracks the following information: the simulation state (in terms of objects and their attributes), actions taken by students and other agents, the location of each student and agent, and human and agent speech (separate messages indicate the beginning of speech, the end, and a semantic representation of its content). In addition, if the agent is tutoring a student, it keeps track of the student’s field of view; messages from the student’s Vista indicate when objects enter or leave the field of view. The cognition module, implemented in Soar (Laird, Newell, & Rosenbloom 1987;

Newell 1990), interprets the input it receives from the perception module, chooses appropriate goals, constructs and executes plans to achieve those goals, and sends out motor commands to the motor control module. The motor control module accepts the following types of commands: move to an object, point at an object, manipulate an object (about ten types of manipulation are currently supported), look at someone or something, change facial expression, nod or shake the head, and speak. The motor control module decomposes these motor commands into a sequence of lower-level messages that are sent to the other software components (simulator, Vista Viewers, speech synthesizers, and other agents) to realize the desired effects. See (Rickel & Johnson 1999) for more details on this architecture.

To allow Steve to operate in a variety of domains, his architecture has a clean separation between domain-independent capabilities and domain-specific knowledge. The code in the perception, cognition, and motor control modules provides a general set of capabilities that are independent of any particular domain. These capabilities include planning, replanning, and plan execution; mixed-initiative dialogue; assessment of student actions; question answering (“What should I do next?” and “Why?”); episodic memory; communication with teammates; and control of a human figure (Rickel & Johnson 1999). To allow Steve to operate in a new domain, a course author simply specifies the appropriate domain knowledge in a declarative language. (Recent work has focused on acquiring the knowledge from an author’s demonstrations and the agent’s experimentation (Angros, Johnson, & Rickel 1997)). The knowledge falls in two categories: perceptual knowledge (knowledge about objects in the virtual world, their relevant simulator attributes, and their spatial properties) and task knowledge (procedures for accomplishing domain tasks and text fragments for talking about them). For details about Steve’s perceptual knowledge, see (Rickel & Johnson 1999); the remainder of the paper will focus on Steve’s representation and use of task knowledge.

#### 3.2 Representing Task Knowledge

Most of Steve’s abilities to collaborate with students on tasks, either as a teammate or tutor, stem from his understanding of those tasks. As the scenario unfolds, Steve must always know which steps are required, how they contribute to the task goals, and who is responsible for their execution. In order to handle dynamic environments containing other people and agents, he must understand the tasks well enough to adapt them to unexpected events; he cannot assume that the task

---

**Task** transfer-thrust-control-ccs  
**Steps** press-pacc-ccs, press-scu-ccs  
**Causal links**  
press-pacc-ccs achieves ccs-blinking for press-scu-ccs  
press-scu-ccs achieves thrust-at-ccs for end-task  
**Ordering** press-pacc-ccs before press-scu-ccs  
**Roles** pacc: press-pacc-ccs; scu: press-scu-ccs

---

Figure 2: An example team task description

---

will follow a pre-specified sequence of steps. Moreover, our goal was to support a declarative representation that would allow course authors to easily specify task knowledge and update it when necessary (Rickel & Johnson 1997b).

Our representation for individual tasks, used in our previous work for one-on-one tutoring, satisfies these design criteria. The course author describes each task using a standard plan representation (Russell & Norvig 1995). First, each task consists of a set of steps, each of which is either a primitive action (e.g., press a button) or a composite action (i.e., itself a task). Composite actions give tasks a hierarchical structure. Second, there may be ordering constraints among the steps; these constraints define a partial order over the steps. Finally, the role of the steps in the task is represented by a set of causal links (McAllester & Rosenblitt 1991); each causal link specifies that one step in the plan achieves a goal that is a precondition for another step in the plan or for termination of the task. For example, pulling out a dipstick achieves the goal of exposing the level indicator, which is a precondition for checking the oil level.

This task representation is suitable for structured tasks based on standard procedures. It would not be suitable for tasks that require creative problem solving, such as design tasks. Fortunately, many tasks in industry and the military have this type of structure, including operation and maintenance of equipment, trauma care, and surgical procedures. Moreover, this representation need not be viewed as a fixed sequence of steps; rather, it is a general causal network of steps and goals, and can be used by a planning algorithm to dynamically order the steps even in the face of unexpected events, as described in Section 3.3.

To extend Steve to team training, we had to decide how to assign team members to task steps. Much of the research on multi-agent teams has addressed methods by which teammates dynamically negotiate responsibility for task steps. However, supporting such negotiation among a team of agents *and* people would re-

---

**Task** loss-of-fuel-oil-pressure  
**Steps** transfer-thrust-control-ccs, ...  
**Causal links** ...  
**Ordering** ...  
**Roles**  
eoow: (transfer-thrust-control-ccs pacc), ... ;  
engrm: (transfer-thrust-control-ccs scu), ...

---

Figure 3: Specifying roles for a subtask

---

quire more sophisticated natural language dialogue capabilities than Steve currently has. Fortunately, many team tasks have well-defined roles that are maintained throughout task execution, so we focus on this class of tasks.

Extending Steve to support such team tasks required one simple addition to each task description: a mapping of task steps to team roles. For example, Figure 2 shows a simplified task model for transferring thrust control to the central control station of a ship. Two roles must be filled: one operator mans the propulsion and auxiliary control console (PACC) in the central control station (CCS), and another operator mans the shaft control unit console (SCU) in the engine room. The PACC operator requests the transfer by pressing the CCS button on her console, which results in the CCS button blinking on both consoles. When the CCS button is blinking on the SCU, the SCU operator presses it to finalize the transfer. This last action achieves the end goal of the task, which is indicated in the task description by specifying its effect as a precondition of the dummy step “end-task.”<sup>1</sup>

If a step in the task is itself a team task, it will have its own roles to be filled, and these may differ from the roles in the parent task. Therefore, the parent task specifies which of its roles plays each role in the subtask. For example, Figure 3 shows a partial description of a task for which the task in Figure 2 is a subtask. This task description calls for the executive officer of the watch (EOOW) to play the role of the PACC operator and for the engine room officer (ENGRM) to play the role of the SCU operator for the transfer of thrust control.

Task descriptions (e.g., Figure 2) specify the structure of tasks, but they leave the goals (e.g., ccs-blinking) and primitive steps (e.g., press-pacc-ccs) undefined. The course author defines each primitive step as an instance of some action in Steve’s extensible action library. For example, the step press-pacc-ccs

---

<sup>1</sup>This representation for end goals is standard in AI planners (Russell & Norvig 1995).

would be defined as an instance of `press-button` in which the particular button to be pressed is `pacc-ccs`, the name of an object in the virtual world. The course author defines each goal by the conditions in the simulated world under which it is satisfied. For example, `ccs-blinking` is satisfied when the simulator attribute `scu-ccs-state` has the value “blinking.” Thus, Steve is able to relate his task knowledge to objects and attributes in the virtual world.

### 3.3 Using Task Knowledge

When someone (e.g., a human or agent instructor) requests a team task to be performed, each Steve agent involved in the task as a team member or instructor uses his task knowledge to construct a complete task model. The request specifies the name of the task to be performed and assigns a person or agent to each role in that task. Starting with the task description for the specified task, each agent recursively expands any composite step with its task description, until the agent has a fully-decomposed, hierarchical task model. Role assignments in the request are propagated down to subtasks until the task model specifies which team member is responsible for each step. For example, if the task is `loss-of-fuel-oil-pressure` (Figure 3), with Joe as the `EOOW`, then Joe will play the role of the `PACC` for the subtask `transfer-thrust-control-ccs` (Figure 2), and hence he is responsible for the step `press-pacc-ccs`. Since all agents have the same task knowledge, each agent will construct the same hierarchical task model with the same assignment of responsibilities.

For simulation-based training, especially for team tasks, agents must be able to robustly handle unexpected events. Scripting an agent’s behavior for all possible contingencies in a dynamic virtual world is difficult enough, but the problem is compounded when each agent must be scripted to handle unexpected actions by any human team member. One option is to simply prevent human students from deviating from standard procedures, but this robs the team of any ability to learn about the consequences of mistakes and how to recover from them. Instead, we have designed Steve to use his task knowledge to adapt task execution to the unfolding scenario.

To do this, each agent maintains a plan for how to complete the task from the current state of the world. The task model specifies all steps that *might* be required to complete the task; it can be viewed as a worst case plan. Agents continually monitor the state of the virtual world, identify which goals in the task model are already satisfied, and use a partial-order planning algorithm to construct a plan for completing the task (Rickel & Johnson 1999). This plan is a

subset of the task model, consisting of the steps relevant to completing the task, the ordering constraints among them, and the causal links that indicate the role of each step in achieving the end goals. In our prior work, this plan would specify how an agent intended to complete a task; for team training, the plan specifies how the agent intends for the team to collectively complete the task, with some causal links specifying the interdependencies among team members (i.e., how one team member’s action depends on a precondition that must be achieved by a teammate). Thus, agents dynamically interleave construction, revision, and execution of plans to adapt to the unfolding scenario.

If an agent is serving only as a missing team member, it simply performs its role in the task, waiting when appropriate for the actions of its teammates, and communicating with them when necessary. In contrast, an agent serving as an instructor for a human student interacts with that student in a manner similar to one-on-one tutoring. The agent can demonstrate the student’s role in the task, explaining each action it takes, or it can monitor the student as she performs the task, answering questions when the student needs help. Moreover, the agent instructor can easily shift between these two modes as the task proceeds; the student can always interrupt the agent’s demonstration and ask to finish the task herself, and she can always ask the agent to demonstrate a step when she gets stuck.

### 3.4 Team Communication

In team tasks, coordination among team members is critical. Although team members can sometimes coordinate their actions by simply observing the actions of their teammates, spoken communication is typically required. Team leaders need to issue commands. Team members often need to inform their teammates when a goal has been achieved, when they are starting an activity, and when they detect an abnormal condition. Because team communication is so important, it must be taught and practiced in team training.

We model team communication as explicit speech acts in the task descriptions. For the sort of structured tasks we have studied, this is natural; all the documented team procedures given to us specified when one team member should say something to another and how it should be said. To support this, we extended Steve’s action library to include a new type of action: a speech act from one team member to another. The specification of the act requires four components: (1) the name of the task role to which the speech act is directed (e.g., `scu`), (2) the name of the attribute being communicated (e.g., `thrust-location`), (3) the value being communicated for that attribute (e.g., `ccs`), and

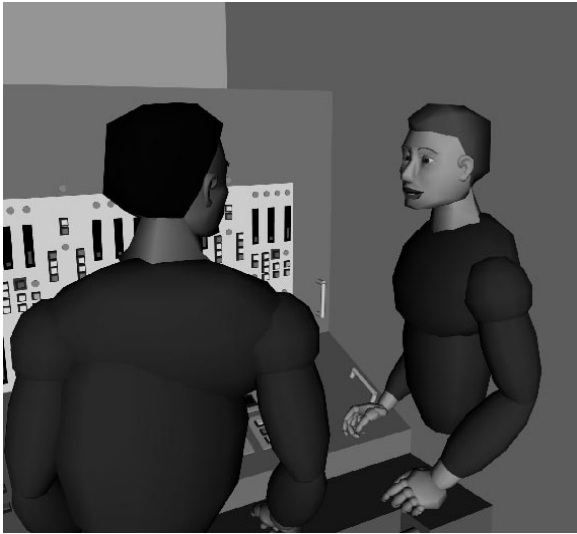


Figure 4: One Steve agent speaking to another

(4) the appropriate text string (e.g., “Thrust control is now at the central control station”). (Note that we do not want to leave this text string up to a natural language generator, because there is often a precise utterance that should be used.) Each speech act appears as a primitive action in the task description, allowing us to explicitly model its relationship to the task, including the role responsible for performing it, ordering constraints on when it should be said, and causal links that specify how its effect contributes to completing the task (i.e., which other steps depend on that result).

Given this representation for team communication, Steve agents can both generate and comprehend such utterances during task execution. When an agent’s plan calls for it to execute one of these speech acts, it sends the text string to appropriate speech synthesizers for its human teammates to hear, and it broadcasts a semantic representation of the speech act for its agent teammates to “hear.” When a human says the appropriate utterance, her speech recognizer identifies it as a path through its domain-specific grammar, maps it to an appropriate semantic representation, and broadcasts it to the agents. Each agent checks its plan to see if it expects such a speech act from that person at that time. If so, it updates the specified attribute in its mental state with the specified value, and it nods to the student in acknowledgment. If the speech recognizer fails to understand the student’s utterance, or the utterance is not appropriate at the current time, the student’s instructor agent is responsible for giving the student appropriate feedback.

There are several important points about this approach. First, it only applies to structured tasks for



Figure 5: One Steve agent watching another

which the required team communications can be specified in the task description; it will not suffice for tasks that require more arbitrary communication. Fortunately, many well structured team tasks, particularly in the military, include such a prescribed set of utterances. Second, since Steve does not include any natural language understanding abilities, all valid variations of the utterances must be added to the grammar for the speech recognizer. Again, this is reasonable for tasks with prescribed utterances. Third, note the difference between our approach and communication messages in a purely multi-agent system; a speech recognizer cannot tell to whom the utterance is intended, so agents use the task model to determine whether the speaker is addressing them. Finally, each agent must treat a human student and their instructor as jointly performing a role; if either of them generates the speech act, it must be treated as coming from that role.

Although spoken communication is typically required for team tasks, nonverbal communication is also important. Human students can observe the actions of their nearby agent and human teammates, which is often required for proper team coordination. Agents look at a teammate when expecting them to do something, which can cue a student that she is responsible for the next step. Agents look at the teammate to whom they are speaking (Figure 4), allowing students to follow the flow of communication and recognize when they are being addressed. Finally, agents react to their teammates’ actions; they look at objects being manipulated by teammates (Figure 5), and they nod in acknowledgment when they understand something a teammate says to them. For tasks that require

face-to-face collaboration among team members, such nonverbal communication is critical.

## 4 Discussion

Steve has been tested on a variety of naval operating procedures. In our most complicated team scenario, five team members must work together to handle a loss of fuel oil pressure in one of the gas turbine engines. This task involves a number of subtasks, some of which are individual tasks while others involve subteams. All together, the task consists of about three dozen actions by the various team members. Steve agents can perform this task themselves as well as in concert with human team members.

Several other recent systems have applied intelligent tutoring methods to team training, although none provides virtual humans like Steve. The PuppetMaster (Marsella & Johnson 1998) serves as an automated assistant to a human instructor for large-scale simulation-based training. It monitors the activities of synthetic agents and human teams, providing high-level interpretation and assessment to guide the instructor's interventions. It models team tasks at a coarser level than Steve, and is particularly suited to tracking large teams in very dynamic situations.

AETS (Zachary *et al.* 1998) monitors a team of human students as they run through a mission simulation using the actual tactical workstations aboard a ship, rather than a virtual mockup. AETS employs detailed cognitive models of each team member (including eye movements, individual keystrokes, and speech) to track and remediate their performance. However, the system does not use these models to provide surrogate team members, and the automated tutor provides feedback to students only through a limited display window and/or highlighting of console display elements.

AVATAR (Connolly, Johnson, & Lexa 1998) provides simulation-based training for air traffic controllers. It monitors and remediates their verbal commands to simulated pilots and their console panel actions. However, the automated tutor and pilots have no planning capabilities, so scenarios must be more tightly scripted than in our approach.

Perhaps the closest work to ours is the Cardiac Tutor (Eliot & Woolf 1995), which trains a medical student to lead cardiac resuscitation teams. The system includes a simulation of the patient, as well as simulated doctors, nurses and technicians that play designated team roles. However, these teammates do not appear as virtual humans; they are heard but not seen. Medical protocols, expressed as linear sequences of actions, play the role of our task descriptions, and the system has some abilities to dynamically adapt proto-

cols to the stochastic simulation and to errors by the student. The representation of task knowledge appears less general than ours, since it is tailored particularly to trauma care. Unlike our system, where any team member could be a student or agent, their system is limited to a single student playing the role of the team leader.

Some important limitations in our system could be alleviated by incorporating recent research results from related areas. To go beyond tasks with prescribed utterances, we could leverage ongoing research on robust spoken dialogue (Allen *et al.* 1996). To handle tasks with shifting roles and unstructured communication among teammates, we could incorporate a more general theory of teamwork (Jennings 1995; Levesque, Cohen, & Nunes 1990; Tambe 1997). To handle tasks that involve simultaneous physical collaboration (e.g., two people jointly lifting a heavy object), we will need a tighter coupling of Steve's perception and body control (Badler, Phillips, & Webber 1993). Although research in these areas is still incomplete, many useful methods have been developed.

Steve agents serving as instructors should provide more information on team activities than they currently do. They should provide running commentary on relevant actions of teammates, which can be difficult because it requires the ability to synchronize verbal descriptions with real-time events in the virtual world. They should indicate perceptual cues that can help the student track teammates' actions. They should interleave their demonstrations of the student's role with descriptions of how and why the student's actions are needed by teammates and vice versa. Our representation of team tasks should support all these capabilities, but they have not yet been added to Steve's tutorial repertoire.

There is a growing understanding of the principles behind effective team training (Blickensderfer, Cannon-Bowers, & Salas 1997; Burns, Salas, & Cannon-Bowers 1993; Smith-Jentsch *et al.* in press; Swezey & Salas 1992). Empirical experiments are beginning to tease out the skills that make teams effective (e.g., task skills vs. team skills), the basis for team cohesion (e.g., shared mental models), the best types of feedback (e.g., outcome vs. process), and the best sources of feedback (e.g., instructor vs. teammate). Because our approach allows us to model face-to-face interaction among human instructors and students and their agent counterparts, we are now in an excellent position to incorporate and experiment with a variety of these new ideas in team training.

## 5 Acknowledgments

This work was funded by the Office of Naval Research, grant N00014-95-C-0179. We are grateful for the contributions of our many collaborators: Randy Stiles and his colleagues at Lockheed Martin; Allen Munro and his colleagues at Behavioral Technologies Laboratory; and Richard Angros, Ben Moore, and Marcus Thiebaut at ISI.

## References

- Allen, J. F.; Miller, B. W.; Ringger, E. K.; and Sikorski, T. 1996. Robust understanding in a dialogue system. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, 62–70.
- Angros, Jr., R.; Johnson, W. L.; and Rickel, J. 1997. Agents that learn to instruct. In *AAAI Fall Symposium on Intelligent Tutoring System Authoring Tools*. Menlo Park, CA: AAAI Press. AAAI Technical Report FS-97-01.
- Badler, N. I.; Phillips, C. B.; and Webber, B. L. 1993. *Simulating Humans*. New York: Oxford University Press.
- Blickensderfer, E.; Cannon-Bowers, J. A.; and Salas, E. 1997. Theoretical bases for team self-correction: Fostering shared mental models. *Advances in Interdisciplinary Studies of Work Teams* 4:249–279.
- Burns, J. J.; Salas, E.; and Cannon-Bowers, J. A. 1993. Team training, mental models, and the team model trainer. In *Proceedings of Advancement in Integrated Delivery Technologies*.
- Connolly, C. A.; Johnson, J.; and Lexa, C. 1998. AVATAR: An intelligent air traffic control simulator and trainer. In *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems (ITS '98)*, number 1452 in Lecture Notes in Computer Science, 534–543. Springer.
- Eliot, C., and Woolf, B. P. 1995. An adaptive student centered curriculum for an intelligent training system. *User Modeling and User-Adapted Instruction* 5:67–86.
- Hill, Jr., R. W.; Chen, J.; Gratch, J.; Rosenbloom, P.; and Tambe, M. 1997. Intelligent agents for the synthetic battlefield: A company of rotary wing aircraft. In *Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence (IAAI-97)*, 1006–1012. Menlo Park, CA: AAAI Press.
- Jennings, N. 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* 75.
- Johnson, W. L.; Rickel, J.; Stiles, R.; and Munro, A. 1998. Integrating pedagogical agents into virtual environments. *Presence: Teleoperators and Virtual Environments* 7(6):523–546.
- Jones, R. M.; Laird, J. E.; and Nielsen, P. E. 1998. Automated intelligent pilots for combat flight simulation. In *Proceedings of the Tenth Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, 1047–1054. Menlo Park, CA: AAAI Press.
- Laird, J. E.; Newell, A.; and Rosenbloom, P. S. 1987. Soar: An architecture for general intelligence. *Artificial Intelligence* 33(1):1–64.
- Levesque, H. J.; Cohen, P. R.; and Nunes, J. H. T. 1990. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, 94–99. Los Altos, CA: Morgan Kaufmann.
- Marsella, S. C., and Johnson, W. L. 1998. An instructor's assistant for team-training in dynamic multi-agent virtual worlds. In *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems (ITS '98)*, number 1452 in Lecture Notes in Computer Science, 464–473. Springer.
- McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, 634–639. Menlo Park, CA: AAAI Press.
- Munro, A., and Surmon, D. 1997. Primitive simulation-centered tutor services. In *Proceedings of the AI-ED Workshop on Architectures for Intelligent Simulation-Based Learning Environments*.
- Newell, A. 1990. *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.
- Rickel, J., and Johnson, W. L. 1997a. Integrating pedagogical capabilities in a virtual environment agent. In *Proceedings of the First International Conference on Autonomous Agents*. ACM Press.
- Rickel, J., and Johnson, W. L. 1997b. Intelligent tutoring in virtual reality: A preliminary report. In *Proceedings of the Eighth World Conference on Artificial Intelligence in Education*, 294–301. IOS Press.
- Rickel, J., and Johnson, W. L. 1999. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*. Forthcoming.
- Russell, S., and Norvig, P. 1995. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall.
- Smith-Jentsch, K. A.; Zeisig, R. L.; Acton, B.; and McPherson, J. A. in press. Team dimensional training. In Cannon-Bowers, J., and Salas, E., eds., *De-*

*cision Making Under Stress: Implications for Individual and Team Training*. American Psychological Association.

Stiles, R.; McCarthy, L.; and Pontecorvo, M. 1995. Training studio: A virtual environment for training. In *Workshop on Simulation and Interaction in Virtual Environments (SIVE-95)*. Iowa City, IW: ACM Press.

Swezey, R. W., and Salas, E. 1992. Guidelines for use in team-training development. In Swezey, R. W., and Salas, E., eds., *Teams: Their Training and Performance*. Norwood, NJ: Ablex. 219–245.

Tambe, M. 1997. Towards flexible teamwork. *Journal of Artificial Intelligence Research* 7:83–124.

Zachary, W.; Cannon-Bowers, J.; Burns, J.; Bilazarian, P.; and Krecker, D. 1998. An advanced embedded training system (AETS) for tactical team training. In *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems (ITS '98)*, number 1452 in Lecture Notes in Computer Science, 544–553. Springer.