Intelligent Simulation-Based Tutor for Flight Training

Emilio Remolina, Sowmya Ramachandran, Daniel Fu, Richard Stottler

Stottler Henke Associates, Inc.
San Mateo, California
{remolina,sowmya,fu}@stottlerhenke.com

William R. Howse

U.S. Army Research Institute
Fort Rucker, Al
howsew@rwaru-emh1.army.mil

ABSTRACT

Today's military flight simulators have dramatically reduced the cost of training by providing cheaper, effective alternatives to training on a real aircraft. However, flight training is still limited by the availability of instructor pilots. The adage "practice makes perfect" is nowhere truer than in the learning psychomotor skills such as flying. Ideally, trainees should be able to practice flying skills on their own to complement instructor-led training. Most flight simulators do not have any automated assessment and tutoring facilities, making them ineffective as self-paced learning environments.

The Army has funded pioneering research on developing automated tutors for flight training, specifically for training initial-entry rotor-wing pilots. An early rule-based system, called the IFT (Intelligent Flight Trainer), monitored trainees' flight performance and provided adaptive coaching. It provided instructional assistance by regulating the challenge level of a flight task, and through overt spoken feedback to inform trainees when they are flying out of range of specified flight parameters. Evaluations showed that while this system was effective in improving flying skills, it was inflexible in terms of it assessment and instruction strategies.

The Army is currently funding research on a next generation automatic flight trainer, called AIS-IFT, that improves upon the IFT. AIS-IFT is designed to be flexible and extensible in terms of assessment and tutoring procedures. A visual authoring tool lets SMEs and course designers modify or create powerful instructional behavior with little programming effort. Whereas the previous effort had the instructional approach embedded deep in the tutoring system, the new approach separate the specific instructional strategies from the ITS infrastructure, thus empowering SMEs and course authors to create a tutor with pedagogy that is customized to their domain.

ABOUT THE AUTHORS

Dr. Emilio Remolina is an Artificial Intelligence research scientist at Stottler Henke Associates, Inc. He received his Ph.D. in Computer Science, specializing in cognitive robotics from the University of Texas at Austin in 2001. His graduate work focused on "map building", whereby an autonomous robot combines sensory information and actions it performs in order to build and localize in a map of its environment. Dr. Remolina's research interest includes intelligent tutoring systems, planning, simulation and common sense reasoning.

Dr. Sowmya Ramachandran is a research scientist at Stottler Henke Associates, Inc. Dr. Ramachandran received her Ph.D. is Computer Science from the University of Texas at Austin. She has a strong background in a wide variety of Artificial Intelligence techniques, including Intelligent Tutoring Systems, and Machine Learning. Her research interests include application of Artificial Intelligence techniques to Education Technology with a focus on addressing motivational, affective, and meta-cognitive issues. Dr. Ramachandran has headed several intelligent tutoring system development efforts for k-12 education and military training. She is currently heading an effort to develop an intelligent tutoring system for training medical teams and another for training information warfare teams.

Dr. Daniel Fu joined Stottler Henke six years ago after earning a graduate degree in computer science. He currently manages two projects: a cultural decision aid tool, and a project to create a wargaming toolset. The first project

employs a case-based reasoning approach to a decision aid that factors cultural influence into the decision-making process. The system consists of a cultural model, authoring and automated analysis techniques for applying the model to specific situations. The second project, called Warcon, aims to create a wargame construction toolset for AF instructors to quickly create wargames for student use. Most important are advanced user interfaces that cater to the user's sophistication level, ranging from simulator parameter editing to complete wargame construction.

Richard Stottler co-founded Stottler Henke Associates, Inc., an artificial intelligence consulting firm in San Mateo, California, in 1988 and has been the president of the company since then. 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 Navy, Army, Air Force and Marine Corps. Currently he is working on OPFOR MOUT Individual Combatant targeting and firing behavior modeling for the Marine Corps and a Combined Arms ITS as part of the US Marine Corps Combined Arms Command and Control Training Upgrade System (CACCTUS). He has a Masters degree in Computer Science from Stanford University.

Dr. William R. Howse, works as a Research Psychologist and Team Leader with the U.S. Army Research Institute's Rotary Wing Aviation Research Unit at Ft. Rucker, Alabama. His experience includes work in psychophysics, educational program evaluation, design and evaluation of training systems, development of performance measures for individual, crew and collective training systems, and development of personnel selection and classification systems. His current activities involve integration of automated and manual data collection processes and development of artificially intelligent adaptive approaches to simulation-based training.

Intelligent Simulation-based Tutors for Flight Training

Emilio Remolina, Sowmya Ramachandran, Daniel Fu, Richard Stottler
Stottler Henke Associates Inc
San Mateo, California
{remolina,sowmya,fu}@stottlerhenke.com

William R. Howse

U.S. Army Research Institute
Fort Rucker, Al
howsew@rwaru-emh1.army.mil

INTRODUCTION

Today's military flight simulators have dramatically reduced the cost of training by providing cheaper, effective alternatives to training on a real aircraft. However, flight training is still limited by the availability of instructor pilots. The adage "practice makes perfect" is nowhere truer than in the learning psychomotor skills such as flying. Ideally, trainees should be able to practice flying skills on their own to complement instructor-led training. Most flight simulators do not have any automated assessment and tutoring facilities, making them ineffective as self-paced learning environments.

The Army has funded pioneering research on developing automated tutors for flight training, specifically for training initial-entry rotor-wing pilots. An early rule-based system, called the IFT (Intelligent Flight Trainer), monitored trainees' flight performance and provided adaptive coaching. It provided instructional assistance by regulating the challenge level of a flight task, and through overt spoken feedback to inform trainees when they are flying out of range of specified flight parameters. Evaluations showed that while this system was effective in improving flying skills, it was inflexible in terms of it assessment and instruction strategies.

The Army is currently funding research on a next generation automatic flight trainer, called AIS-IFT, that improves upon the IFT. AIS-IFT is designed to be flexible and extensible in terms of assessment and tutoring procedures. A visual authoring tool lets SMEs and course designers modify or create powerful instructional behavior with little programming effort. Whereas the previous effort had the instructional approach embedded deep in the tutoring system, the new approach separate the specific instructional strategies from the ITS infrastructure, thus empowering SMEs and course authors to create a tutor with pedagogy that is customized to their domain.

This paper will describe both these efforts in detail and discuss avenues for future research and development in the area of automated flight training.

AUTOMATED FLIGHT TRAINING

The Army Research Institute (ARI) has been studying the problem of building automated tutors for training initial-entry rotor-wing (IERW) pilots. Normally these pilots are trained in real equipment one-on-one by instructor pilots. The ARI has the objective of improving the efficiency of this process by providing intelligent simulator-based tutors to replace some of the live equipment training. While computer-based training cannot replace live training, it has been shown to be a highly effective complement, especially when it provides scenario-based instruction with realistic simulators [Schank, 1995]. The effectiveness of such simulation-based training hinges crucially on the availability of instructional support from an instructor.

Unfortunately the financial and human resources are simply not available to provide the kind of one-on-one instruction that learning to fly helicopters requires. The use of an intelligent tutoring system (ITS), as the one described in this paper, have the potential to achieve many of the same benefits as one-on-one instruction do. Intelligent Tutoring Systems are computer-based training systems that mimic human instructors in providing one-on-one instruction. Much like a human instructor, ITSs dynamically assess and diagnose a student's knowledge and skill levels and provide training that is customized to the student's learning needs. To truly tailor instruction, ITSs create, develop, and maintain a *model of the student*. This model is used as a basis for automatic selection of instruction method and content, for automatic diagnosis, remedial course formulation, re-testing, progress monitoring and reporting.

IFT: AN INITIAL APPROACH TO INTELLIGENT FLIGHT TRAINING.

The Intelligent Flight Trainer (IFT) is an Intelligent Tutoring System for IERW pilots. The IFT consists of a helicopter flight simulator and an intelligent tutoring system (ITS) merged into a single system. This system

was designed to help teach hovering skills to IERW pilots, and is described in greater detail than found here in previous papers (Krishnakumar, et. al., 1991). The skills taught by the IFT were later extended to include hover taxi, hover turn, traffic patterns, and standard approaches (Mulgand, et. al., 1995). An introduction to helicopter piloting, as well as detailed descriptions of the tasks above, can be found in Padfield (1992).

The IFT simulator represents a generalized training helicopter. The cockpit consists of a frame, instrument panel, cyclic, collective, and pedals, all of which have been taken from actual helicopters. There is also a single screen to display virtual or glass cockpit displays. Three larger screens provide forward and side visual displays, with a resulting visual field of about 90 degrees. The entire simulator (including the ITS discussed below) is powered by a set of Linux boxes. These machines run various pieces of software to control the cockpit, flight model, image generation, and audio systems (Mulgand, et. al., 1995).

In the IFT, the intelligent tutoring system attempts to provide the same types of training provided by instructor pilots. The two main components of the ITS are referred to as the *helper* and the *advisor*. *Helper* makes it easier for the student to fly the helicopter, akin to training wheels on a bicycle. It dynamically adjusts the flight model to correspond with the student's ability to complete maneuvers. The student begins with a flight model that is very easy to fly, but very unrealistic, and progresses to an aerodynamic model that closely approximates the real thing. This allows beginning students who, for example, tend to make large, impulsive cyclic movements, to be able to "fly" the helicopter. At the same time, proficient students are not given this freedom and need to make the small and controlled types of cyclic inputs actually used in the helicopter. All of this is performed without explicit interaction of the student with the ITS.

The second component requires more interaction since the *Advisor* communicates verbally with the student. Currently, this means that the advisor "talks" using text-to-speech software and the student listens. The advisor has four different informative roles, the first of which is to instruct the student on basic procedures (tutorial role), such as applying left peddle as the collective is increased. Performance monitoring is the second role, with instructions such as "watch your airspeed". The third role is control activity monitoring, where comments on how the student is using the controls are given by the advisor (e.g. notifying the student when they are cross-controlling). The final feedback role is advisory, which verbalizes suggestions to control or correct flight. An example of this type of comment is "descend by lowering the collective."

Mulgand, et al., (1995) evaluated the performance of the IFT with a single participant with a basic knowledge of helicopter flight but minimal flight experience. They found that the level of control assistance given could effectively allow the student to hover, and that the level of control assistance generally decreased with more time spent on the hover task. For the traffic pattern task this level of increased performance was not found. Generally, the student performed poorly on this task. They note, however, that the *advisor* did successfully guide the student through the traffic pattern. Despite the evaluation shortcomings (e.g., small study size, lack of control group), the evaluation does serve as an indicator that variable controls can help a student perform tasks and that the student can follow the verbal cues of the *advisor.*

We are primarily concerned with the intelligent tutoring system portion of the IFT. The IFT system was found to somewhat brittle with respect to its instructional approach which was embedded deep into the logic of the system and there were no facilities to change its behavior without re-writing the system. Studies with students showed that its behavior did need to be tuned and modified based on observations of its effectiveness. The current work was therefore motivated by the need to develop an Adaptive Instructional System (AIS) architecture that provides the infrastructure for deep student modeling (i.e. modeling factors other than skill mastery, e.g. personality and affective traits), and for rapid modification and enhancement of tutoring behavior.

The next sections present the AIS-IFT system. We give a system functional description, followed by a description of the system architecture, the authoring tool, and actual content.

AIS-IFT FUNCTIONAL DESCRIPTION

AIS-IFT is a framework for building agile Intelligent Tutoring Systems that can be adapted in terms of their instructional behavior with minimal effort. An overarching goal is to develop approaches for putting the tools for modifying the content and the pedagogy of an ITS in the hands of the trainers and subject matter experts who can incorporate the lessons they learn from extended use of the ITS back into the design of the ITS. An important pre-condition for meeting this objective successfully is that the tools should expose the underlying structure and behavior of the ITS

comprehensively, and provide easy yet powerful ways of modifying them. The trade-off between usability and power is crucial.

The AIS-IFT follows the instructional model, as used by Instructor Pilots, of teaching in the context of flight exercises. Trainees are assigned flight exercises based on mastery and personality attributes. Exercises are preceded by pre-flight briefings, and followed by detailed after-action reviews. The after-action reviews include pointers to remedial material on certain principles; however most of the training happens in the context of exercises. Trainees are coached through the exercises to varying degrees based on their expertise level.

Adaptive instructional practices

The AIS-IFT system incorporates the following adaptive instructional practices:

1. The tutor selects instructional goals based on the student model, which is in turn based on assessment of student performance on exercises. Advanced tasks and skills are introduced only when their pre-requisites have been mastered. Thus, the challenge level of learning activities presented to students are customized to their pace of gaining expertise.

2. The tutor adapts to personality differences by presenting introverts initially with exercises that make fewer attentional demands. This is based on the research finding that introverts are less able to handle situations that require paying attention to several things at once than extraverts. By regulating the challenge level to start with exercises requiring less attention, and progressively increasing this over time, the tutor provides introverts with a graded learning experience that is tuned to their abilities.

3. Expert students are given less coaching than novice students. This is consistent with the recommendation that the optimal strategy for teaching motor skills is to provide coaching and practice in the initial stages but withdraw them as the student progresses. Such withdrawal of coaching has been found to be essential for effective learning [Gagne and Medsker, 1996].

4. The coaching provided to a student is itself adaptive of the student's mastery profile. During a flying exercise, the tutor has several competing dimensions on which it can focus its coaching. For example, at any given time in an exercise, a student may be going above the target altitude and moving away from the target heading. In this case, the tutor uses the student model to choose its focus. For a student who has less mastery over maintaining heading than altitude, it would coach the student on the former. On the other hand a student who is better at maintaining heading than altitude will be coached on maintaining altitude. In this way, the tutor adapts its coaching to the needs of the student.

An AIS-IFT session

Instruction is based on flight exercises. An exercise consists of a flight pre-brief, followed by a flying session, ending with an after action review. Next we describe what an AIS-IFT session looks like.

Assessment testing

For first-time training, the student proceeds to assessment testing followed by a pre-brief, exercise, post-brief, and an optional review. Returning students proceed directly to a pre-brief based on earlier assessments and recent training exercises. The initial assessment currently includes a personality test; the AIS-IFT system evaluates student responses and determines the appropriate learning style and personality of the student based on the answers given by the student in the assessment test.

Posting teaching goals

Following the assessment testing, the AIS-IFT system plans further instruction by developing a set of goals and finding a plan to satisfy those goals, as explain later. For example, a list of goals for a beginner student may include

- (increase-mastery (concept = control cyclic to the left)
- (increase-mastery (concept = decrease altitude))
- (increase-mastery (concept = increase speed))

indicating those skills the student should practice next. The tutor prefers active exercises to passive display of didactic review and therefore most often chooses to assign exercises as a way of improving skills. When selecting an exercise, the tutor choses that exercise that cover the most goals.

Pre-brief

Each exercise starts with pre-brief providing instructions of what the exercise is about and what is expected from the student (Figure 1). The pre-brief provides a specific task outline instructing the student of step-by-step action and any requirements that must be met during the exercise, such as helicopter heading, altitude and speed requirements in an exercise. Task's descriptions include hyperlinks that if selected, provide additional information and detail for guidance and definition of the task. This pre-brief is customized

depending on the student's model and learning style. For example, if students have some familiarity with the task less information about the task is provided; for some students the task is presented in a well structured and detailed manner; for other students the task is succinctly described and more links are provided to present opportunities for analysis and research.

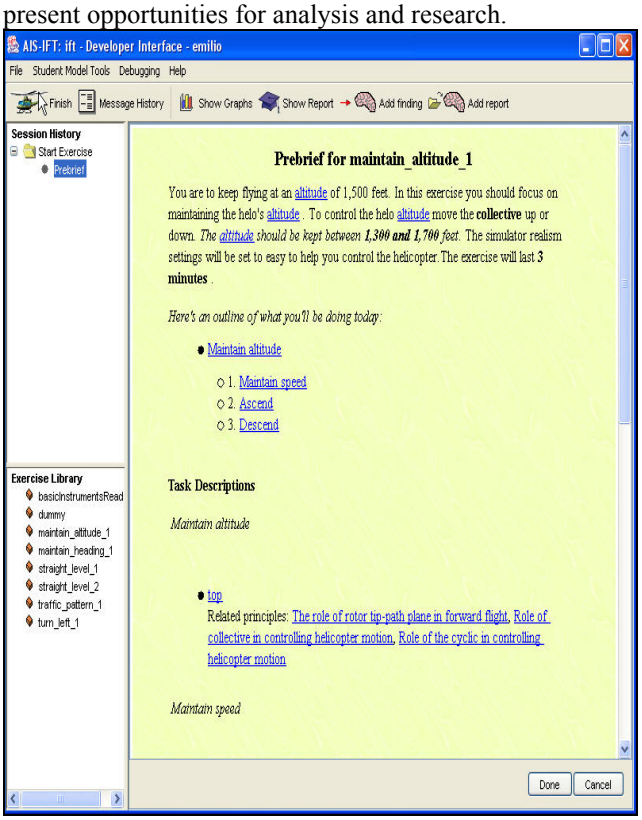

Figure 1. Exercise pre-brief.

Doing the exercise

To begin an exercise, AIS-IFT instructs the student to select a specific flight that is geared for the chosen exercise. The student uses the simulator facilities to start the flight. Once the flight starts, the student is in control of the helicopter. The tutor provides (spoken) instructions whenever it decides the student needs some help. The tutor decides and informs the student when the exercise is done.

Real-time coaching feedback during the exercise is derived from a description of the procedure the student should carry on to recover from out of nominal conditions (e.g., if the student lost altitude use a recover altitude procedure to coach the student). In later sections we illustrate how authors can "draw" such procedures.

The tutor provides the following kinds of coaching depending on the expertise level of the student. Advanced students get limited coaching, which is often restricted to alerting the student about events and helicopter conditions that need attention. Novice students, on the other hand, get hands on coaching in the form of specific instructions on what they should be doing with the controls. Coaching takes the form of verbal, spoken instructions. In addition, the tutor may provide help in the form of **visual cues**. In the current version of AIS-IFT, the tutor can flash relevant instruments to guide the student in using instruments to understand the state of the helicopter and determine corrective actions. For example, if the student is climbing too fast, the tutor will flash the *climb-rate indicator* in order to draw the student's attention to her rate of climb.

The tutor stops the simulation whenever the student loses control of the helicopter. Losing control of the helicopter happens when the helicopter's parameters are outside the exercise's specified range, which is specified while defining the exercise using the authoring tool. Usually these parameters define obvious out of control conditions: the helo is about to crash, the helo is rolling. Other out of control conditions are less obvious: the helo deviated too much from the exercise's targeted heading; the helo is out of the altitude range specified for the exercise. The tutor will explain why the helo is out of control, as well as how to correct the situation.

Post-brief

The tutor provides an after-action review once the exercise is completed. This feedback is given in two forms: an exercise performance summary (see figure below) and a replay of the exercise.

The postbrief is a typical exercise performance summary that shows the following results: (i) the three best things that were done well; (ii) improvements (if any) noticed in student's flying skills (none shown in this figure), and (iii) three worst things done during the exercise. Hyperlinks are provided for the student to review those flying skills or principles that need the most improvement. In all cases, the student model is used to filter the feedback that is provided to student by not including things the student already knows (e.g., point out things done well only if the student has not mastered them before the exercise) or things that are not usual problems (e.g., advanced students may lost altitude during an exercise although they in general know how to maintain the altitude).

After-action review also includes a graphical summary of the exercise execution. For example, in a straight-level exercise, these graphs show the change in latitude, longitude altitude, and heading during the

exercise execution. This review is set so students can see the changes as they occurred.

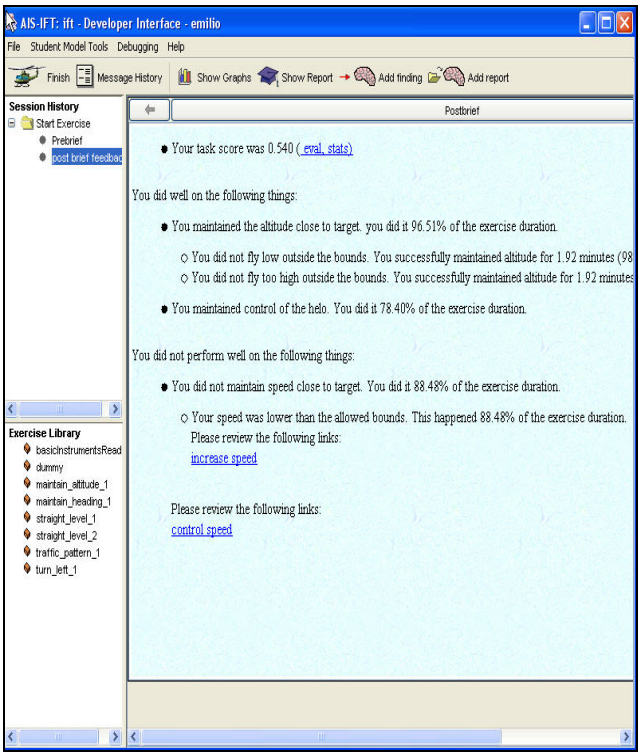

Figure 2. After action review display

Handling automaticity

Developing skill automaticity is very important for achieving expertise in motor tasks such as flying. Automaticity is the degree to which a person has automated the skills so that they require minimal attentional resources to perform them. People who have automated a skill can typically perform other secondary tasks without any deterioration of their performance on tasks involving the skill. Most of us have automated the skill of driving a car so that we are able to converse with passengers or listen to the radio without significant loss of driving performance. Automaticity is typically assessed by observing a student's performance on the main task in the presence of other distractions that require their attention. The section *student assessment and modeling* explains how the student model has been designed to include automaticity measures.

AIS-IFT measures automaticity by compounding flight exercises with secondary tasks that compete for a trainee's attention. Currently, the AIS-IFT includes one secondary task, namely manipulating various radio control buttons that are used to assess automaticity. Secondary tasks can be graded according to their difficulty so that more challenging distractions can be used as the student gains expertise.

The student's performance on the flight task in the presence of distractions is measured and used to update the student model. These exercises not only serve as assessments but also as practice opportunities that let students to develop automaticity.

AIS-IFT ARCHITECTURE

AIS-IFT consists of two components: an authoring tool and a domain-independent ITS run time engine. This core runtime engine requires content in the form of presentations, domain ontology, and assessment and instructional behaviors which must be created using the authoring tool.

Figure 3 shows the relationship between the domain content (mostly defined via the authoring tool) and the core components of the runtime engine: the *Assessment Manager*, the *Simulation Manager*, the *Communications Manager*, the *Instructional Planner,* and the *Sensor Manager*. The box with the thick solid line defines the boundaries of the core parts of AIS. The components within the box with the dashed boundary represent AIS-IFT components that are domain independent, but still can be swapped in and out. For example, it should be possible to replace a personality assessment test without modifications to the core system. The components outside the two boxes represent the domain dependent components and will have to be developed separately for each domain of application (e.g., the simulator, or special sensors like eye tracking). The architecture of the runtime engine and its interfaces has been designed so that these domain-specific elements can be simply incorporated as plug-ins. Next we discuss the assessment manager and the instructional planner. The other engine components mostly do data transformation and facilitate the communication between the engine and domain-dependent components

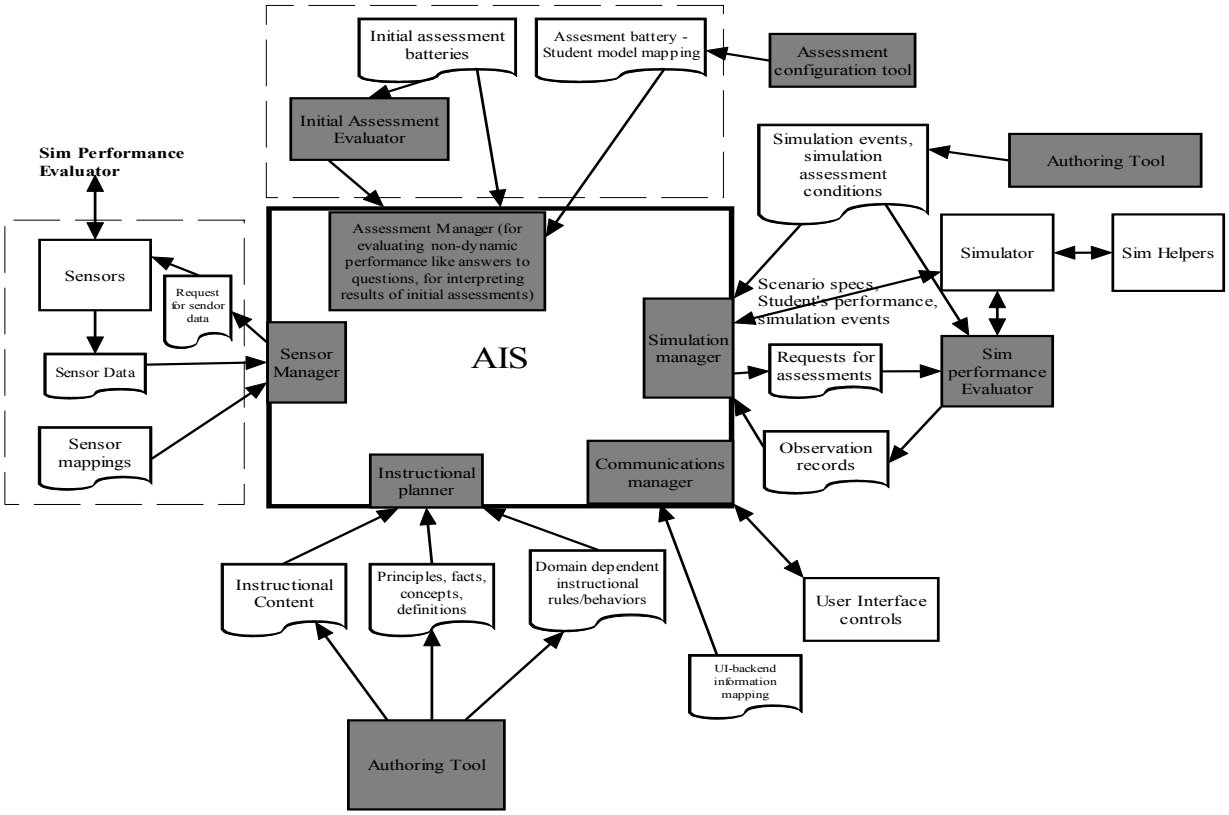

Figure 3. AIS-IFT system architecture.

Student Assessment and modeling

The *Assessment manager* uses the domain ontology specified by the course authors as the basis of the student model. The domain concepts such as tasks, skills, and principles are augmented with numeric estimates of mastery and, in the case of tasks and skills, automaticity. The assessment of a student's performance in exercises provides the basis for these estimates. Bayesian inference is used to integrate the findings from a single exercise with the existing student model estimates. The Assessment manager uses the student model to decide which skills the student should work next. This information is converted into instructional goals posted to the instructional planner.

Figure 4 shows an example of a Bayesian network constructed automatically from the relation between tasks (hover requires maintain altitude), skills (collective control, pedal control) and principles (maintain sight of picture, not shown in the figure). The mastery nodes represent the student's mastery on a task, skill, or principle without accounting for automaticity. The automaticity node models the degree to which the student shows evidence of having automated a task or a skill. The "true mastery" node represents a composite of these two estimates.

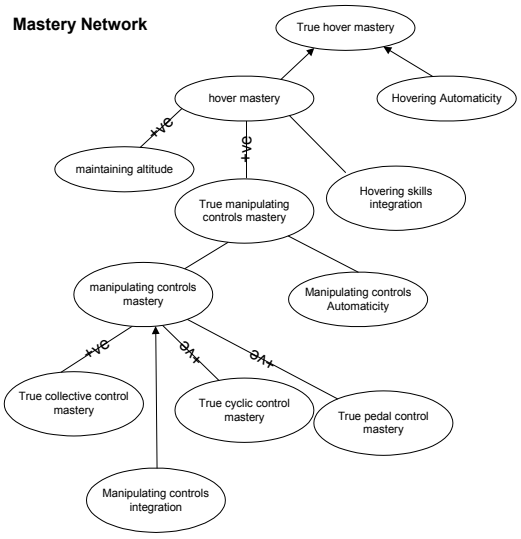

Figure 4. Bayesian network used to represent the student model.

In the absence of any initial assessment, the Bayesian network is instantiated with uniform priors that indicate low mastery on all domain concepts. The conditional probabilities representing the relationship

between skills and their parent concepts are also heuristically instantiated. When the *Simulation Performance Evaluator* reports on the student's performance on a simulated exercise, it also sends scores on the various tasks, skills, and principles associated with the exercise. These scores are used to heuristically determine the evidence of mastery on these concepts and are placed as evidence on the corresponding nodes in the Bayesian network (mastery evidence are placed on mastery nodes, and evidence from automaticity assessments are placed on automaticity nodes). Bayesian inference is used to propagate this evidence to arrive at updated mastery estimates (Pearl, 1986).

Instructional planner

The Instructional Planner (IP) decides what instructional action the system should do next. The IP (i) creates and executes *plans* to satisfy posted *goals*, (ii) provides facilities to broadcast event/observations to *agents*, and (iii) provides facilities to relate agents to goals, to plug-in new agents, and to execute hierarchical exercises.

Making plans

In order to satisfy a goal, the IP asks all agents whether they can satisfy that goal. Based on the agents' answers and the planning strategies (e.g., prefer doing simulation exercises to reading review material), the IP creates a plan. Each step in the plan is an agent's action. The IP executes this plan by asking agents to execute the action associated with the current step of the plan.

Modeling Agents

At the heart of the AIS-IFT engine there is a collection of *agents* responsible for carrying out instructional actions. Agents know how to do certain *actions*. An action is associated with a *behavior*, which is the "executable" part of the action. Actions can be thought of as "what to do" and behaviors as "how to do it."

On being asked to perform an action, an agent "activates" the *behavior* associated with the action. Agents can run multiple behaviors simultaneously. Although running simultaneously, behaviors share the same thread of execution. Behaviors are modeled as C++ objects defining a method called *action* which implements the logic of the behavior. A round-robin non-preemptive scheduling policy executes the *action* method of each active behavior until the behavior releases control (the behavior's *done()* method return true). If the behavior relinquishing the control has not yet completed (the behavior's *done()* method returns false), it will be rescheduled the next round. This form of modeling agents is similar to the one used by systems like JADE which follows FIPA's standards (www.fipa.org).

When agent's behaviors are described using the authoring tool, SMEs are not aware of the agent's API described in the previous paragraph. Authors define behaviors by drawing hierarchical state machines akin to flowcharts, and the AIS-IFT follows such description (see authoring tool section below). Nevertheless, the system APIs allow developers to enhance agent's behaviors by using different application tools like rule-based systems (e.g., CLISP) or Bayesian networks (e.g., Netica).

AIS-IFT AUTHORING TOOL

The authoring tool allows authors to define declarative and procedural knowledge. Declarative knowledge defines a database of object instances, each consisting of a *type* which is a set of properties common to a number of instances that distinguish them as an identifiable class. Types and instances define a semantic network representing the domain of the ITS. The authoring tool GUI organizes this database in terms of the following editors (see Figure 5): Task-skill-principle Editor, the Exercise Editor, and the Student Model Editor. The *Task-skill-principle* editor enables the definition of the knowledge of what to teach and includes the following default types of knowledge objects: tasks, skills, and principles. These define the core set of domain knowledge. The *Exercise* editor facilitates the creation of a library of exercises for the tutor to draw upon as it trains the students. The *Student Model* editor defines the attributes that should be modeled in the Bayesian network used to define the student model. The GUI provides facilities to browse, query and perform consistency checks on the domain database.

AIS-IFT exercises were represented by instances of the type *Exercise*. Exercise's attributes represent among others the following information:

- A collection of tasks, skills, and principles that describe what the exercise teaches.
- A set of pre-requisites the student must satisfy in order to do the exercise.
- Exercise parameters: a list of values for the variables of interest during the exercise (e.g., target altitude, target speed, maximum speed deviation).
- Evaluation machines: behaviors describing procedures the student is expected to perform during an exercise.

- Evaluator: behavior in charge of producing the evaluation report when the exercise finishes.
- Exercise coordinator: behavior describing when different stages of the exercise should start or end.
- Events: list all the events (detected conditions) the tutor should react to.

The author can extend the type hierarchy to represent exercises that need special representation. For instance, in a traffic pattern exercise it is necessary to know the *waypoints* at which the student is supposed to turn.

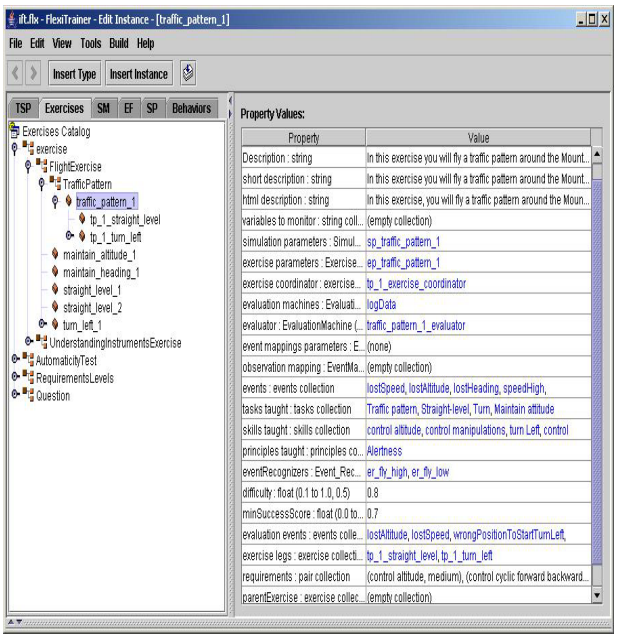

Figure 5. Different editors (tabs) allow authors to define a database of types and instances used to represent the instructional domain.

Procedural knowledge - Modeling behaviors

Types and instances provide a way for gathering knowledge. Ultimately, there are two ways in which the knowledge will become operational: evaluating and teaching the student. The ways in which the training system fulfills these functions are driven by behavior scripts that dictate how the training system should interact with the student. The *Tutor Behavior* editor has the author specify two kinds of knowledge: how to assess the student and how to teach the student. Both types of knowledge are captured in the form of behavior scripts that specify tutor behavior under different conditions. These behaviors are visualized in a "drag and drop" style canvas (see Figure 6).

AIS-IFT models behaviors as a hierarchical finite state machine where the flow of control resides in stack of hierarchical states. Condition logic is evaluated according to a prescribed ordering, showing very obvious flow of control. AIS-IFT employs four constructs: *actions*, which define all the different actions the system can perform; *behaviors* that chain actions and conditional logic; *predicates*, which set the conditions under which each action and behavior will happen; and *connectors*, which control the order in which conditions are evaluated, and actions and behaviors take place. These four allow one to create behavior that ranges from simple sequences to complex conditional logic. The graphical representation of a behavior is akin to a flowchart as illustrated in the figure below. Next we present some of the AIS-IFT teaching and evaluation procedures.

Teaching an exercise

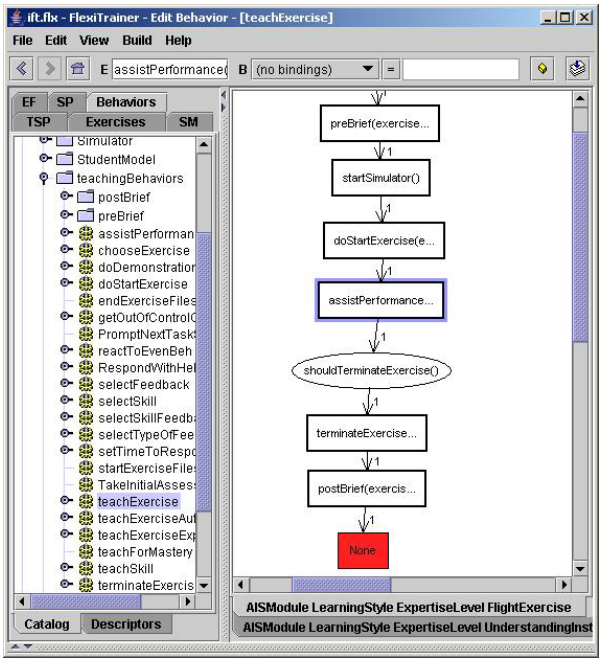

Figure 6. Behavior defining the high-level logic on how to teach an exercise.

Figure 6 shows the *teachExercise* behavior defining how to teach a flight exercise. This behavior will present an exercise prebrief, start the simulator, coach the student during flying and then give the student after action review. The behavior also defines places at which the student may cancel the exercise: during the prebrief or while executing the exercise. The behaviors used by teachExercise have the following function:

- **prebrief**: speaks aloud the short description of the exercise, generates an html file describing the exercise, and shows this html file to the student.

- **startSimulator**: verifies that the simulator is connected (running) before starting the exercise. If the simulator is not connected, it asks the student to start the simulator.
- **doStartExercise**: sets the initial conditions for the scenario: starts the simulator with the correct scenery file and place the helicopter in stable conditions as described by the exercise parameters.
- **assistPerformance**: this behavior is in charge of providing real-time feedback to the student during the exercise. (more later)
- **postBrief**: provides after action review to the student: it generates an html file with a summary of things done well, any improved skills, and things to improve. In the later case links to review material are provided. The student will also have access to a replay of the exercise.

Evaluation Machines – Events

AIS-IFT uses the following schema to evaluate student performance in real time and provide coaching feedback during an exercise:

1. *Evaluation machines* generate *events* for which some feedback may be provided.
2. The *assistPerformance* behavior takes as input events generated by evaluation machines and decides on which event to pay attention to (if any) and which feedback to provide (if any).
3. The exercise coordinator will start and stop the evaluation machines associated with each exercise's stage so that appropriated events (and so feedback) are generated.

Events signal the occurrence of some condition during the exercise. For example, an event may signal that the helicopter is flying too high, while another event may signal that the student did not move the collective down as expected. Events are generated by evaluation machines.

Evaluation machines monitor whether the student is following some procedure. When monitoring a procedure, the context for whether an event be reported is defined by the current state of the procedure. Consequently, events are context-dependent, and the feedback associated with these events will be consistent with the execution of the evaluation's machine flying procedure.

Figure 7 shows the high level description of the task maintain altitude: the student is expected to monitor the altimeter while maintaining constant altitude. Should the altitude start increasing (altimeterTooHigh condition in the figure) the student should start a descend procedure to acquire altitude.

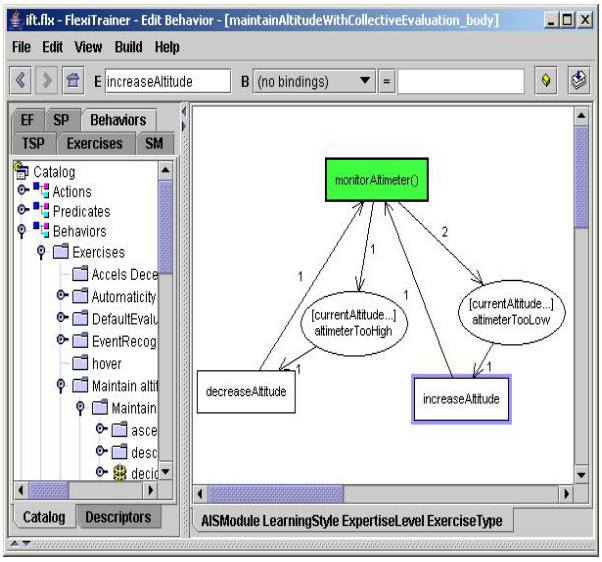

Figure 7. High-level description of a maintain altitude procedure.

The logic of the *decreaseAltitude* behavior is as follows (see Figure 8):

1. Wait for student to do something. If student moves the collective down (decreasedCollective condition in the figure), start the real descent procedure (box *labeled by doDescendWithCol*).

2. If the student does nothing and some time has elapsed (*timeout* condition, left upper corner in the figure), then post an event (*descendingNoCollectiveDown*) and get ready to start the real descent procedure. If the tutor decides to react to this event, using the event's feedback, the tutor will ask the student to move the collective down.

3. If while waiting for the student's action the altitude gets larger than the safety largest altitude, and *outofcontrol* event is generated.

The lower part of Figure 8 shows the rest of the behavior's logic, which is concerned with the condition to end the behavior *doDescendWithCol.* This logic is as follows:

4. If the helo gets stable at the target altitude, the procedure ends.

5. If while descending, the target altitude is overshot, then the behavior *doAscendWithCollective* is invoked.

6. If during ascend the helo overshoots the altitude, then a behavior is generated indicating overshoot from below and the descend procedure is invoked again.

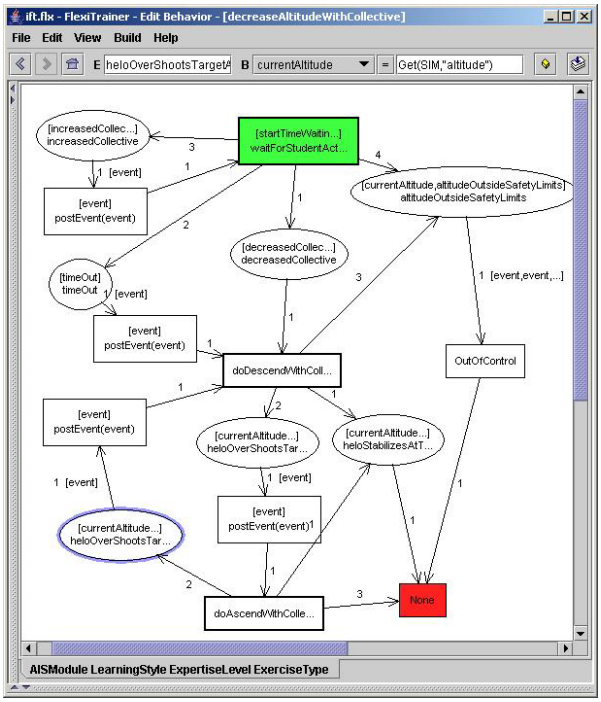

Figure 8. Procedure to decrease altitude.

The behavior *doDescendWithCo* describes the procedure by which the student acquires target altitude. The procedure recognizes three basic helicopter states: descending, ascending and ascending above target altitude. Here is the logic of this procedure:

1. If the helo is descending but the descend rate is too high, then the event *descendingTooQuicklly* is generated.
2. If the helo has been ascending for the last 5 seconds, then an event is generated (whose feedback will make the helo go down), and we wait for 2 seconds before deciding in which of the three main states the helo is at. This wait recognizes the fact that it takes some time before observing the effect of a control input, and consequently, the tutor should not ask the student to provide more input than required.
3. If the helo is stable above target altitude, behave similarly as when the helo is ascending. The event generated may have the same feedback (i.e., nose down) but signal different conditions for the procedure and the types of problems the student may have.

CONCLUSIONS AND FUTURE WORK

Simulation-based flight training augments with intelligent tutoring to enable automated training has the potential for delivering effective training at reduced costs. We have discussed a series of such systems, the IFT and AIS-IFT. The latter improves upon the former by enabling rapid modification of the content, the assessment procedures, and instructional strategies. AIS-IFT exposes not only the content but also tutoring behavior to inspection and modification. The authoring tool provides a visual metaphor resembling flow-charts to describe desired tutoring and assessment procedures. Helicopter flying procedures are described using this tool and at run time the system monitors that the student follow such procedures, coaching the student if needed by providing feedback that is appropriate according to the procedure's current state and context. The AIS-IFT is one of the early steps in the direction of automated flight training. Evaluation studies will have to be conducted to study its performance and fine tune the training methodology.

The current version of AIS-IFT runs on a desktop using Microsoft flight simulator. Our future work involves integrating the ITS with the IFT simulator as well as evaluating the system, both the performance of the ITS and the authoring tool facilities. While very powerful, the current authoring tool requires authors to have some programming skills to exploit the whole range of possibilities.

ACKNOWLEDGEMENTS

The work reported here was funded by the Office of the Secretary of Defense under contract number DASW01-01-C-5317.

REFERENCES

Krishnakumar, K.S., Sawal, D., Bailey, J.E., & Dohme, J.A. (1991). A simulator-based automated helicopter hover trainer: Synthesis and verification. In *IEEE Transactions on Systems, Man, and Cybernetics, 21,* 961-970.

Mulgand, S.S., Asdigha, M., Zacharias, G.L., Krishnakumar, K., & Dohme, J.A. (1995). An intelligent tutoring system for simulator-based helicopter flight training. In *Proceedings of the 1995 Flight Simulation Technologies Conference.*

Padfield Randall R. and Padfield Ralph C. 1992. *Learning to fly helicopters*. McGraw Hill.

Pearl, J. (1986), *Probabilistic reasoning in intelligent systems: Networks of plausible inference*, Morgan Kaufman.

Schank, R. (1995), *What We Learn When We Learn by Doing*, Technical Report no. 60, Institute of Learning Sciences, Illinois.