Principle Hierarchy Based Intelligent Tutoring System for Common Cockpit Helicopter Training

Robert A. Richards

Stottler Henke Associates, Inc. (SHAI)
1660 So. Amphlett Blvd., Suite 350
San Mateo, CA 94402, U.S.A.
Richards@shai.com, www.shai.com

Abstract. SHAI is developing a comprehensive Operator Machine Interface Assistant (OMIA) system. The system will assist operators learn the new common-cockpit MH-60R and MH-60S helicopters in an increasingly broad variety of mission tasks and analyses, using the wide assortment of sensor, navigation, and computational resources available. We are employing a principle hierarchy based intelligent tutoring system, to teach the Common Cockpit machine interface during simulated mission execution. The Intelligent tutoring system utilizes a SHAI developed ITS authoring environment that has been developed and enhanced for various other ITS projects. This paper describes the OMIA system, emphasizing the authoring tool's ability to develop visually the principle hierarchy and the evaluation finite state machines necessary to evaluate the student. In addition the decision tree enhancement to the authoring tool developed as part of this OMIA project, is described.

1 Introduction

The US Navy is introducing two new helicopters, the MH-60S and MH-60R (See Fig. 1.). Both of these helicopters utilize the *Common Cockpit* design. The Common Cockpit includes all the flight and mission instrumentation in both of the helicopters and enables both the pilot and co-pilot to share workload through dual flight and mission instrumentation, see Fig. 2. SHAI is building a training tool called the Operator Machine Interface Assistant (OMIA) that includes an intelligent tutoring system (ITS) to teach [1] the common cockpit. OMIA is currently in use by the US Navy and is being expanded to teach more of the overall domain.

Fig. 1. MH-60R

OMIA utilizes a principle hierarchy based approach to the intelligent tutoring that interfaces to a scenario-based free-play simulation [2]. Much of the intelligent tutoring system aspects of the overall OMIA training system have been developed utilizing an ITS authoring tool (ITSAT) developed by SHAI. In addition, OMIA has

Fig. 2. Common Cockpit

required capabilities beyond the scope of the previous version of ITSAT and has thus enhanced ITSAT in a general manner to meet OMIA's needs.

2 OMIA Training System

Two major components of the OMIA System are the ITS, and the scenario-based free-play simulator. The simulator allows the student to learn via free-play scenarios with a graphical user interface that closely matches that of the common cockpit mission displays. As shown in Fig. 3 the simulator communicates with the ITS, (via an external system interface) to update the student model, and to provide the student with remediations.

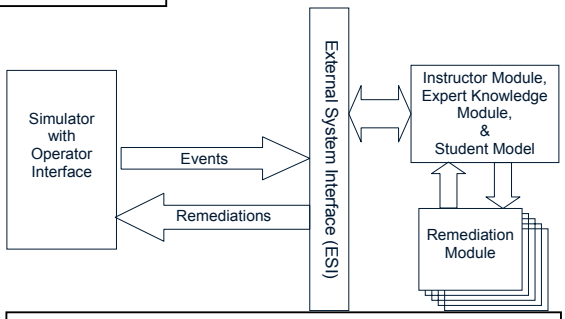

Fig. 3. OMIA Major Components

2.1 Simulator

The simulator provides an engaging interface between the student and the ITS. A student interacts with the simulator through a computer re-creation of the controls onboard the MH-60R/S, see Fig. 4. In addition to being an interface, this component also simulates the environment surrounding the helicopters using *scenarios*. Scenarios are authored using a visual editing tool, and determine the elements of the simulation. For example, sonar returns from pinging depend not only upon the settings chosen by the operator, but also on the environment and target submarine settings in the scenario

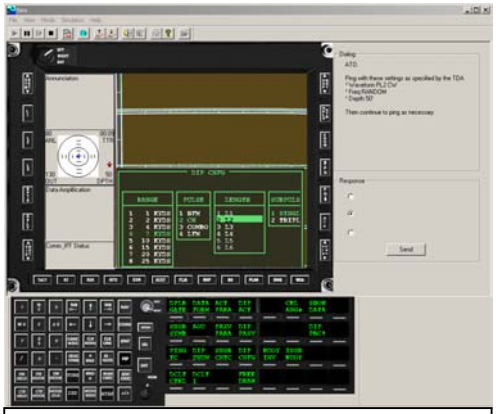

Fig. 4. Simulator's Operator Interface

file. Other entities, such as an enemy submarine or a friendly ship, can have agendas of their own. For instance, a submarine can be assigned the behavior "flee on detection." Even the fleeing behavior itself is customizable, as are all behaviors.

Of course, an operator might add objects such as sonobuoys to the simulation in real time. The flip side is that destroyed/sunk objects would be removed from the simulation as well. The ability to act upon the simulation and have it respond gives the students freedom to do the right thing, or to make mistakes. Either way, the system then has a better model of the student than before, and can use this to improve his learning experience. A graphical scenario generator [3] allows an instructor/author to create complex scenarios that include any number of intelligent agent entities (e.g., ships, submarines, and aircraft).

2.2 ITS Design

The OMIA ITS consists of the following major modules:

student model,
instructor,
expert knowledge, and
remediation.

This paper concentrates on the principle hierarchy and the tools used to manipulate the principle hierarchy. More information regarding other aspects of OMIA may be found in [4].

The Expert Knowledge Module utilizes a *principle hierarchy* representation, i.e., a collection of individual principles, arranged in a hierarchy. The principles are relatively low-level pieces of testable information. Each principle may also contain material that should be presented to the student as remediations. A very small sample of the principle hierarchy is shown below.

- Acoustic
  - Active Sonar
    - Dipping Sonar
      - Unintegrated for fast targets
      - Waveform selection
      - Configure before ping
      - Ping after configure

The lower levels of the hierarchy are composed of specific principles which include the operational knowledge necessary to determine the appropriate action for a crewman to take in a particular set of circumstances. As events unfold, various principles come into play suggesting the appropriate conclusions and reactions resulting from new data and circumstances. The principle hierarchy, and associated knowledge about proper application of principles to scenarios, provides a facility for detecting at any time in a mission

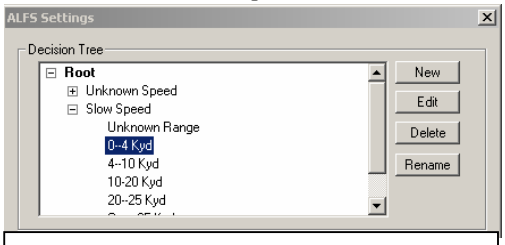

Fig. 5. Dipping Sonar Decision Tree

a correct action (e.g., evasive maneuvers) and/or determining a correct conclusion (e.g., target submarine is deeper than expected).

An example of an individual principle is "Use the unintegrated setting on the dipping sonar for fast moving targets." In this case, comparing a student's actions to the expert's is straightforward. A slightly more complicated principle is "Correct waveform selection for the dipping sonar." For this principle, the sonar settings suggested by the expert vary depending upon the environment and the expected target. A *decision tree* is used to represent the domain expert knowledge for principles such as this.

Decision trees are graphically constructed tree diagrams where at each node a question is asked. The next node is chosen based on the answer to the current question. By traversing through this tree, one eventually arrives at the correct decision. A partial decision tree for determining dipping sonar settings based on submarine presumed speed and distance is shown in Fig. 5 and Fig. 6. In Fig.5 the highlighted *0 – 4 Kyd* is a leaf of the tree for a slow moving target at a range of 0 to 4 kiloyards. For this leaf the proper settings are shown in Fig. 6. In this situation, there are two dipping sonar configurations that are appropriate.

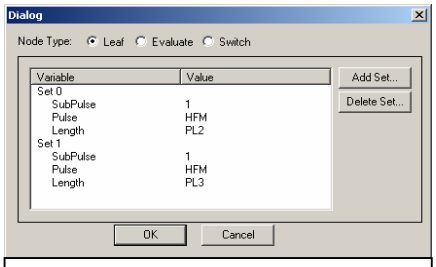

Fig. 6. Dipping Sonar Decision Tree Example Settings

3 Using the Intelligent Tutoring System Authoring Tool (ITSAT)

There are six categories of knowledge in the OMIA ITS system. These are;

- The principle hierarchy.
- The simulation scenarios, to be used as examples and exercises.
- Descriptions (multi-media) which explain each principle.
- Knowledge used to asses the correctness of student actions.
- Knowledge used to assess a student's mastery of a principle given the history of his performance in relation to that principle.
- Pedagogical knowledge.

SHAI has developed tools to assist in the development of all of these categories of knowledge. The SHAI developed ITS Authoring Tool (ITSAT) allows ITS authors to organize course principles, articulate teaching methods, specify courseware, and develop a case base of scenarios for students along with a specification of how the student's actions will be evaluated and their mastery of the required knowledge assessed. This paper demonstrates primarily the visual ITSAT environment with regards to the principle hierarchy and the finite state machines that contain the knowledge used to asses the correctness of student actions, i.e., they evaluate the student's knowledge of the principles.

3.1 Principle Hierarchy Editor

ITSAT's principal hierarchy screen is shown in Fig. 7. The *Hierarchy* tab has been developed and evolved during the course of many SHAI ITS projects [5] to provide information about the principle hierarchy and the constituent principles and remediations as well as to provide a visual method of creating, manipulating and destroying principles and remediations. Each principle can be associated with multiple elements from the remediation section.

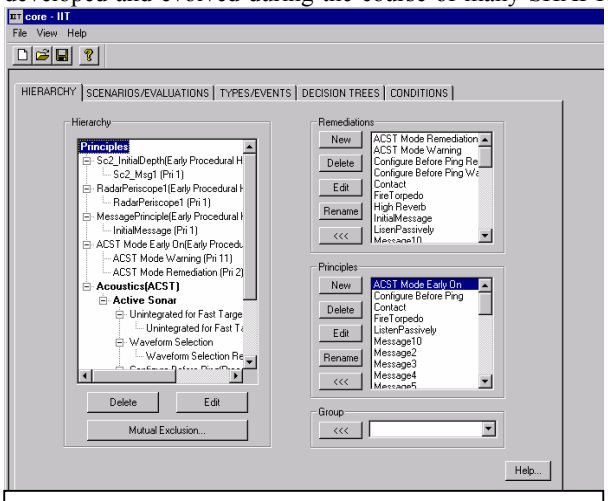

Fig. 7. ITSAT – Main Screen Showing Hierarchy Tab

The left **Hierarchy** region shows the principle hierarchy and is used to organize the principles into the hierarchy itself. The bold entries in the hierarchy are groups, e.g., "Acoustics(ACST)". These groups represent areas of knowledge that encompass one or more principles. Under the groups are the principles, and under each principle are the principle's remediations. Objects in the hierarchy may be edited via a double click or the Edit button, while the Delete button removes items from the hierarchy.

Similarly, the **Remediations** region and the **Principles** region provide buttons to easily create, delete, edit, and rename items; as well as add items to the hierarchy (via the <<< key). The **Group** region provides the mechanism to create and add new groups to the principle hierarchy.

For example, there is a visual layer for creating or editing remediations, as shown in Fig. 8. In the remediation dialog, the *remediation type* indicates the type of this remediation file. This information determines how the file will be presented to the student. Types include, multimedia (i.e., video), text, picture, and default. In the case of *default* OMIA will ask Windows to open and display the file. One can test the remediation via the *Play* button; this will show how the remediation is presented to the student.

Fig. 8. Remediation Editor

3.2 Instructor, Student Model, and Remediation Modules

It is the job of the instructor module to determine when a student has failed a principle or conversely when a principle has been successfully applied. If for example the expert module indicates that a student should search for short range targets before long range targets, but the student does the opposite, he has failed this principle. The instructor module would then annotate the student's *student model*, and inform the remediation module, which may provide immediate feedback to the student (remediation) if appropriate.

The OMIA ITS contains *evaluation finite state machines* (EFSM) that describe what principles are active in a given situation, and evaluate a student's performance during the execution of a scenario. That is, it is the evaluation machines that are watching the actions of the student and determining which principles are being performed correctly or incorrectly. A scenario can have any number of evaluation machines that are evaluated simultaneously. ITSAT provides a visual tool for constructing, managing and editing the EFSMs.

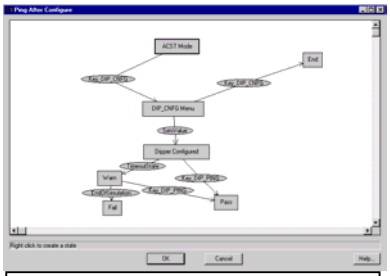

Fig. 3. Evaluation Finite State Machine Editor

A graphical representation of an EFSM is shown in Fig. 9. After a student performs an action, for example pinging the dipper, a different EFSM might check the student's waveform selection against that of the expert waveform selection. If the settings are correct, the student's percentage correct on waveform selection would increase. If wrong, his percentage on this principle would decrease and the remediation attached to this principle might be displayed.

EFSMs consist of states and transitions. Each state has a set of transitions; each transition has a destination state. During evaluation, the EFSM is considered to be in one of its states, called the "current state." From this current state, the EFSM evaluates each of the state's outgoing transitions. If one of these transitions "becomes true", it is taken to its destination state. This state then becomes the "current state" of the EFSM. Additionally, the student passes any of the transition's passing principles, and/or fails any of the transition's failing principles. This structure creates a set of directional paths that are used to determine how well the student reacted to events in the scenario.

3.3 Evaluation Finite State Machine (EFSM) Editor

ITSAT includes an evaluation machine editor. In the main window of the editor one can graphically create and connect states and transitions using the mouse. The text box below this window displays hints for editing the EFSM. The basic operations include:

- To create a state: Move the mouse over a blank portion of the window, and then right click. A box representing the state will appear.

- To create a transition: First select a state to be the source state. Then move the mouse over the state that will be the destination state for the new transition. Finally, right click. A transition will be created going from the source state to the destination state. The transition object itself is represented graphically as an oval.
- To select a state or transition: Move the mouse over the state or transition and click.
- To edit a state or transition: Move the mouse over the state or transition and double click.
- To delete a state or transition: Select a state or transition, and then press the delete key.

The descriptions button brings up the description editor for the evaluation machine that can be used to provide documentation.

As an example, in the EFSM shown in Fig. 9, the initial state, *ACST Mode*, is the top-most rectangle. When the action, *Key_DIP_CNFG* is taken, a transition is made to the *DIP_CNFG Menu* state. From this state, the machine can either transition on the dipper being configured (SetsValue) or on the key which closes the menu without changing any settings (Key_DIP_CNFG).

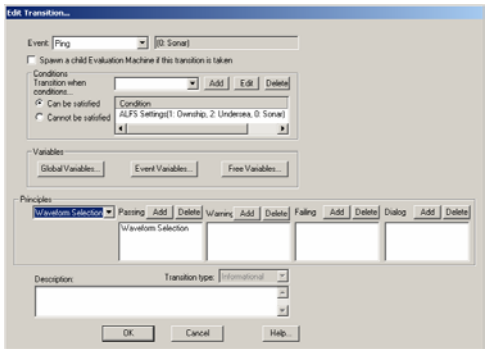

Fig. 10. Evaluation Transition Editor

A major portion of the information content for an EFSM is contained within the transitions. Transitions can be activated based on either an *event* or a *condition*, or both. For example, a transition might be triggered on the event *OpenScenario* with a condition of *Scenario name equals Scenario Bravo*. ITSAT provides a GUI to work with the transitions and is further described below.

3.3.1 Evaluation Transition Editor

The evaluation transition editor is shown in Fig. 10. Each transition has a list of conditions. In general, when these conditions are satisfied, the transition "becomes true" and is taken by the evaluation machine. A simple or complex set of factors can determine whether a transition is taken or not.

The event on which to match for the transition is set in the *Event* field. If an event is specified, this transition will only be taken if the event occurs AND all the conditions are met. If "None" is specified, the transition is taken any time the conditions are met. To the right of this field are the bindings of the event parameters. These event variables become bound to the particular event that occurred, and can be used as parameters to the transition's conditions.

If the *Spawn a child Evaluation Machine if this transaction is taken* box is checked, then the evaluation machine will remain in its current state and spawn a copy of itself when the transition would normally be taken. The spawned evaluation machine will begin evaluating at the transition's terminal state. This technique is useful for creating evaluation machines that need to evaluate a situation that can happen multiple times simultaneously.

The **Conditions** region displays all the current conditions for this transition. The drop-down list contains all possible conditions that can be added. If the "Can be satisfied" radio button is selected, then the transition will only be taken if all the conditions can be satisfied. If the "Cannot be satisfied" radio button is selected, then the transition will be taken only if all the conditions cannot be satisfied. Buttons are provided to; *Add* the selected condition to the current list of conditions, giving it default parameters; *Edit* the parameters of the condition via a Condition Editor; and *Delete* the condition from this transition. OMIA has enhanced the previous version of ITSAT to include the capability to have decision trees as one of the conditions. See below for more information on decision trees.

The **Variables** region allows editing of the different types of variables in a transition. These variables can then be used as parameters to the transition's conditions. Variables have a type, which indicates what type of object they can refer to, types include *Global, Event* and *Free.* Global variables are global in the sense that they are shared between the states and the transitions of the evaluation machine. Changes to the value of a global variable made in one transition will be seen by all transitions in that evaluation machine. Event and free variables are local variables. That means that they are only available in the context of the current transition. Free variables are declared as part of a transition, and the system will assign them to any object that causes the conditions to be satisfied. Event variables are parameters that are tied to an event. ITSAT includes variable editors for all types of variables.

The **Principles** region allows for adding/removing passing/failing principles for this transition. When a transition is taken, the student is credited with passing all the "passing" principles indicated, and is considered to have failed all the "failing" principles. To attach a principle to the transition, simply select it in the drop-down list. Then, choose to add it to one of four categories: passing, warning, failing, and dialog. When the transition fires, an action will be performed based on the category of the attached principle(s). For *passing* principles, a success will simply be noted in the principle hierarchy. *Warning* principles indicate that if the principle has a warning associated with it, then this warning should be displayed to the user. In the OMIA system, this is known as an enhancement and shows up on the multi-function display. A warning of *TryCOM1 PL4* is shown in Fig. 11. If a *failing* principle is attached it is noted that the principle was failed in the principle hierarchy.

Fig. 11. Operator Interface with Warning

The *transaction type* field specifies what kind of transition this is. This provides feedback in the evaluation summary as to whether this transition is purely

informational, whether it means some event occurred, or whether the student performed some action that caused this transition to be taken. In any case, this field is only applicable for transitions in which no principles are passed or failed, and it has no impact on the evaluation.

The *description* field annotates the transition with comments that may be displayed to the student after they're evaluated, i.e., in an ex postfacto summary report. The *comment* field is used to describe the transition in plain English, e.g., to provide explanation to other authors.

3.4 Decision Tree Editor

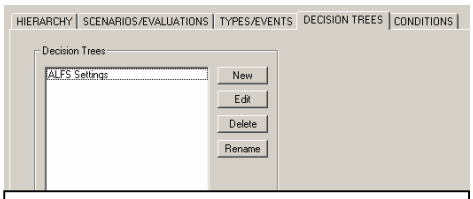

Fig. 12. Decision Tree Tab

OMIA's major contribution to the ITSAT tool is the addition of decision trees. A decision tree (DT) provides a powerful new type of event transition condition. The complex environment in which the MH-60S and MH-60R helicopters operate requires judgments that must take into account many variables. In addition, there are situations where multiple answers are correct. For multiple answer situations EFSMs could be constructed to handle them, but handling them in the context of the DT proved much cleaner.

The OMIA development could have built the DT capability outside the context of ITSAT, however, it was determined early that the power of DTs is formidable and should be included as a key component of ITSAT. The Decision Tree Editor follows the look and feel of the rest of ITSAT. The *Decision Tree* tab in ITSAT, as shown in Fig. 12, provides the basic mechanisms for creating *New* decision trees, as well as the ability to E*dit*, *Delete* and *Rename* decision trees.

Fig. 13. Edit Decision Tree Dialog

The dialog accessed via the *Edit* button, see Fig. 13, begins to reveal the power of the decision tree editor. All DTs have a **Root** node, from the **Root** node one may add (*New)* nodes, as well as *Edit, Delete,* and *Rename* nodes. This dialog also allows for the selection of arguments available to the nodes and the return values.

There are a series of dialogs that allows for the creation of nodes, nodes can be one of three types, Leaf, Evaluate, or switch. An example of a complete leaf node is shown in Fig. 14.. This is the Leaf node for the Unknown Speed, Unknown Range branch of the decision tree shown in Fig. 13.

The decision tree editor addition to the ITSAT tool has proven valuable to the OMIA ITS development, both in the development of decision trees and in the maintenance and understanding of them.

4 Conclusion

The complexity and number of the sensors under control of the crew on the MH-60S and MH-60R helicopters pose a difficult training task for the Navy. To meet this challenge SHAI is developing a comprehensive Operator Machine Interface Assistant system that employs a principle hierarchy based intelligent tutoring system. SHAI has exploited its own rapid development intelligent tutoring system authoring tool to construct the principal hierarchy, the evaluation finite state machines, decision trees and other portions of the ITS. The intelligent tutoring system authoring tool's capabilities have continued to expand as more and more SHAI ITS projects have found the tool valuable and then enhanced the tool to add any new capability required be particular ITS projects. The enhancements have been added under the same general framework so that added power and flexibility have come with only a small increase in user complexity. OMIA is no exception, the decision tree capability has been added during OMIA's development process. OMIA, benefiting from the power of ITSAT, has proven its value to the US Navy, as it is currently in use to train MH-60S crew as further development continues.

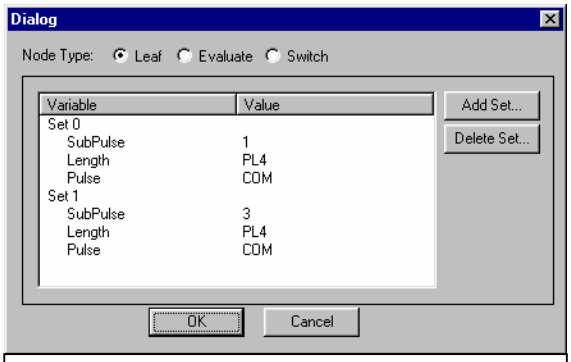

Fig. 14. Decision Tree Node Dialog

References

1. Bloom, B. S., (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. Educational Researcher, 13(6): 4-16.
2. Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.) (1999). How People Learn: Brain, Mind, Experience, and School. Washington D. C.: National Academy Press.
3. Stottler, R. H., & Vinkavich, M. (2000). Tactical action officer intelligent tutoring system (TAO ITS). I/ITSEC 2000 Proceedings.
4. Ludwig, Jeremy L. & Henry Jackson (2001). A Common Cockpit Training System. I/ITSEC 2001 Proceedings.
5. Stottler, R. H., Fu, D., Ramachandran, S. & Vinkavich, M. (2001). Applying a Generic Intelligent Tutoring System (ITS) Authoring Tool to Specific Military Domains. I/ITSEC 20001 Proceedings.