Simulation-Based Automated Intelligent Tutoring

Barbara Sorensen[1] and Sowmya Ramachandran[2]

[1] Air Force Research Laboratory, Mesa, AZ USA
Barbara.sorensen@mesa.afmc.af.mil
[2] Stottler Henke Associates, Inc., San Mateo, CA USA
Sowmya@stottlerhenke.com

Abstract. Simulation-based training has traditionally relied on live simulations for the specification of training scenarios. However, both the logistics and cost of live simulation constrain how often that form of training can be accessible. As the healthcare community begins to invest significantly in simulation-based training, it is running against the limitation of requiring expensive hands-on instructor-led facilitation to make it effective. New technological advances applied to simulation-based training provide automated coaching and feedback which reduce the need for instructor facilitation, thus making simulation-based training cost-effective and feasible. The design and development of SimCore (**Sim**ulate, **Co**ach, **Re**view), a simulation based framework that adds intelligent performance assessment and coaching facilities to training simulations, is discussed. Through the use of SimCore, simulation developers are able to convert their simulations into automated, intelligent tutors. SimCore is a unique application that includes a scenario authoring tool tailored to the medical domain that can be used to rapidly customize existing scenarios or to create new ones.

Keywords: Simulation-based training, Intelligent Tutoring, Scenario Authoring Tool.

1 Introduction

Simulation-based training can significantly benefit learners by providing them the opportunity to practice and learn from applying their skills and knowledge to realistic training scenarios [1,2,3]. Traditionally, simulation-based training has meant live simulations. The logistics and cost of such simulation constrain how often they can be presented. Typically such simulations deliver one or two scenarios. With computer-based simulations, it is possible to deliver cost-effective training with the added benefit that trainees can play them anywhere, anytime. In order to support such self-paced learning, simulations must be accompanied by performance assessment and feedback. As the healthcare community starts to invest significantly in simulation-based training [4], it is running against the limitation of requiring hands-on instructor-led facilitation to make it effective. This is prohibitively expensive. Automated coaching and feedback reduces the need for instructor facilitation, thus making simulation-based training cost-effective and feasible.

M.J. Smith, G. Salvendy (Eds.): Human Interface, Part II, HCII 2007, LNCS 4558, pp. 466–474, 2007.

According to Kolb [5], experiential learning is a cyclic process where a student works with a concrete situation, reflects on his experience, creates abstractions of the knowledge gained from the experience, and finally tries his new knowledge on other related situations. Simulations provide the concrete experience. Without reflection and abstraction, the experiential learning cycle would be incomplete. Typically instructors work with students to help complete the learning cycle; however, this makes the cost of simulation-based training considerably higher. Additionally, providing a personalized learning experience requires one-on-one instruction which can increase the cost substantially. Techniques from the fields of Artificial Intelligence and Intelligent Tutoring systems can be applied to automate one-on-one personalized instruction including automated performance assessment, coaching and review. Incorporation of these techniques will close the loop on the experiential learning cycle while freeing simulations from the requirement for dedicated instructors.

We are currently developing technologies for applying the concept of Intelligent Tutoring Systems (ITS) to healthcare simulations. ITSs are software tutors that are designed to provide one-on-one tutoring, much like humans [6]. As a first step in this direction, we are developing SimCore (**Sim**ulate, **Co**ach, **Re**view), a tool that adds intelligent performance assessment and coaching facilities to training simulations. Using SimCore, simulation developers can convert their simulations into automated, intelligent tutors. Freed from the need for hands-on instructor-led support, simulation systems can be used to deliver not one but a variety of scenarios, giving students the opportunity to learn by applying their newly acquired skills under varying conditions. SimCore also allows for complex free-play simulations. It can be used to actively guide a student towards performing an action or a procedure, even as the simulation allows them the freedom to explore. Authoring tools are typically needed to make ITs a viable option [7]. SimCore includes a scenario authoring tool that can be used to rapidly customize existing scenarios or to create new ones, thus, precluding the additional time and expense of acquiring a compatible authoring tool.

2 Architecture

SimCore employs a client-server architecture with two server-side components (Figure 1): the authoring tool and the runtime server. The authoring tool enables the specification of training scenarios in abstract, high-level terms. The runtime server drives the simulation, initiates events, responds to and evaluates student actions, and provides after-action reports. The system has been designed to interface with third party simulators with minimal customization of the runtime engine or the authoring tool. A default Flash-based simulator is included with the SimCore package.

The runtime server connects to a simulator via a simulator Application Programming Interface (API). This simulator API must be customized to work with each specific Simulator. For the included simulator, communication between the runtime server and the simulator is performed using Extensible Markup Language (XML). In general, the simulator is responsible for maintaining the user interface for one or multiple players. The runtime server is responsible for maintaining the state of the simulated world.

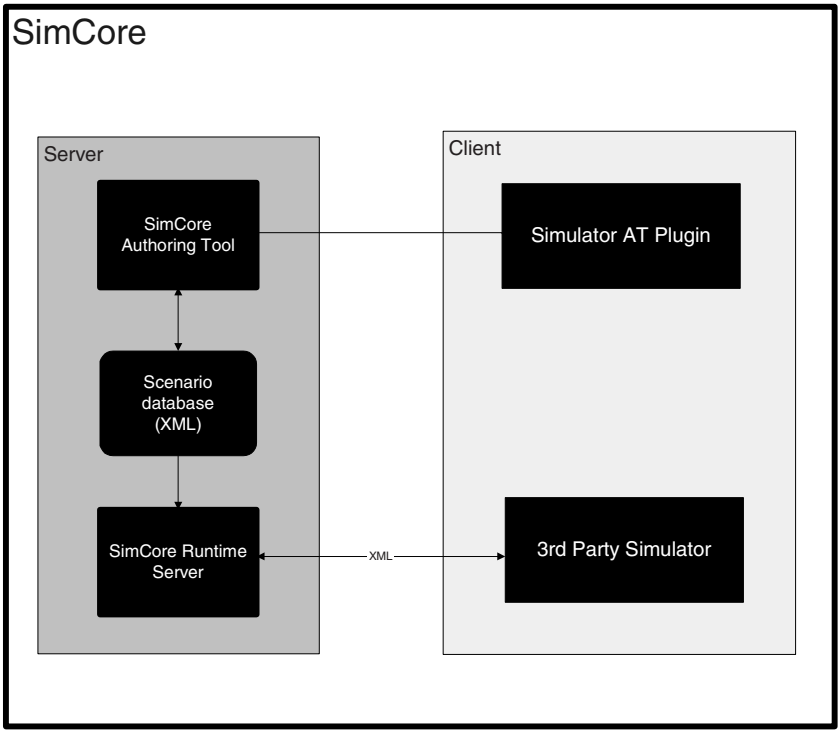

Fig. 1. SimCore client-server architecture with two server-side components

In a typical case, with the SimCore Runtime Server running, the player would run the Simulator. The Simulator would have a list of scenarios from which the player can choose. The list of scenarios is generated by the Runtime Engine. After selecting a scenario, the player uses the Simulator to start the simulation. The Simulator notifies the Runtime Engine when the simulation begins, and the Simulation Engine sends packages of information that contain the current location of the player and other locations the player can move to, all of the props in the location, and all of the actions that can be performed on those props or on the location itself.

The Simulator receives this package of information and is responsible for organizing these options for the player. The Simulator is responsible for displaying a representation of the current location; whether text based, 2D graphical or 3D graphical. In general, the player has two options: perform an action on a prop (or location) or move to a new location. Once the player chooses one of these options, the Simulator sends a message to the Simulation Engine containing the option chosen.

Upon receiving a move message, the Simulation Engine moves the player to the new location. It calculates the props and the actions that can be performed on these props and packages them up into a new message to be sent to the Simulator. When this message is sent to the Simulator, the Simulator acts just as when the game started: it displays a representation of the new location, and sets up a mechanism for the player to access all of the possible actions for the new location.

Performing an action on a prop (or at a location) is the core mechanism of the simulation. The action is tied to the item or location and, performing an action potentially changes the state of the simulation world.

In the Runtime Engine, there are a series of events that are triggered by actions or other events. These events do things like change the simulation state, set up timers (for triggering other events,) end the simulation, etc.

Eventually, an event or a performed action will cause the simulation to end. At this point, the Runtime Engine evaluates the player's performance and sends a message to the Simulator that contains the score data. The Simulator displays this data and the simulation ends.

3 Scenario Model

SimCore is based on a generalized, domain-independent scenario model. In the runtime server, the actions, props, etc. are represented by a set of Java Objects. These building blocks are created by the authoring tool, stored in a series of XML files, and read in by the runtime server. A scenario is composed of locations, props, NPCs (non-playing characters), and events. Locations represent the various different logical places in a scenario. For example, a side of the street which is the scene of an accident can be a location. An ambulance may be a second location. Props are scenario objects in locations and can have actions performed on them. The car involved in an accident is an example of a prop. NPCs are virtual human characters in the scenario. The person hurt in the accident, as well as his companions in the car would be modeled as NPCs. Props and NPCs are associated with attributes that represent their states at any given time. For an NPC, for instance, the modeled attributes may include his/her vital signs, age, medications, allergies. The scenario model includes actions that be performed on the NPCs and Props.

Each action has a unique command name, a textual representation to display to players, a series of possible textual replies upon performing the action, a series of sub-areas on the target where the action can be performed, a set of preconditions as to when this action can be performed, and a set of effects that are executed after the action is performed. For example, “Check Blood Pressure” may be an action with the precondition that the blood pressure monitor should be available for performing this action. The action definition would also include rules specifying the simulators response when the player performs this action. A scenario also includes events. Sometimes it is necessary to change the scenario state when the player does not initiate an action. Events are the way to accomplish this. Events are a set of effects that will be performed if a set of preconditions are met. They also contain a trigger set that, if equal to true, will cause this event's preconditions to no longer be checked and the event will be made inactive. An example is a box (prop) that is supposed to explode in 30 seconds if it has not been opened. An event is created that has a precondition that checks to see if 30 seconds has passed. If so, run the effect that causes the box to explode. The inactive section of the event checks to see if the box has been opened. If so, then the event's preconditions will no longer be checked, and

the box won't explode because of this event. Authoring a scenario entails instantiating the scenario model. The Runtime Server executes this model to deliver a training scenario.

4 Automated Performance Assessment and Coaching

Built specifically to support human interaction with computers, an automated performance assessment component was developed and included in SimCore. This performance assessment component also guides students towards an optimal solution path. Each scenario can be associated with a solution template using the SimCore authoring tool. A solution template is a generalized procedure or protocol that is required to complete the scenario successfully. For the purpose of understanding the solution template, a generalized procedure, is one that can include unordered sets of actions and conditional actions. This solution template is not visible to the player; it is used behind-the-scenes to assess student performance. The solution template defines the actions the student should perform. In the development of the scenarios, the scenario author can also specify specific error rules that represent actions the student should not perform. This capability allows for more in depth scoring and feedback to the player.

During each scenario, the simulator software sends notification messages to the SimCore engine that describe each student action and reports the values of each simulation state variable. SimCore evaluates each student action by comparing it with the solution template and the error rules contained within the scenario definition. This information is passed back to the simulation which then shows the appropriate feedback. In addition, this information is included in the after-action review. The solution template is also used by SimCore to provide just-in-time hints which are associated with the solution template during authoring.

5 Authoring Tool

The authoring tool is a very important component of the SimCore framework. It enables the rapid development and modification of training scenarios. The authoring tool has multiple, tailored, visual editors for defining the scenario model described earlier. These editors include domain specific “wizards” which guide the author in creating the building blocks of the scenario. In addition, there is a section in the authoring tool for specifying the solution templates and error rules. Hints and feedback to help coach can also be specified using these editors. Figure 2 below shows the Props editor where scenario author can create Props, define their attributes, and the actions that can be performed on them.

To easily understand and participate in the building of a scenario, it can be broken down into the following parts: (1) designing the physical world, (2) populating the world, (3) adding events to the world, (4) linking the scenario world to the simulator Graphical User Interface (GUI), and (5) keeping track of what the student is doing.

Designing the physical world entails laying out a series of locations for a world map. Where can the players go? Are there locations that are unreachable at the start of the scenario? After establishing the locations, items or Props are added to the

scenario. Props are inanimate objects such as cars, trees, houses or anything that the players can interact with in the scenario. If the player does not need to interact with an object, it does not need to be defined as a Prop. In addition, there are special Props called Tools that can interact with Characters in the scenario.

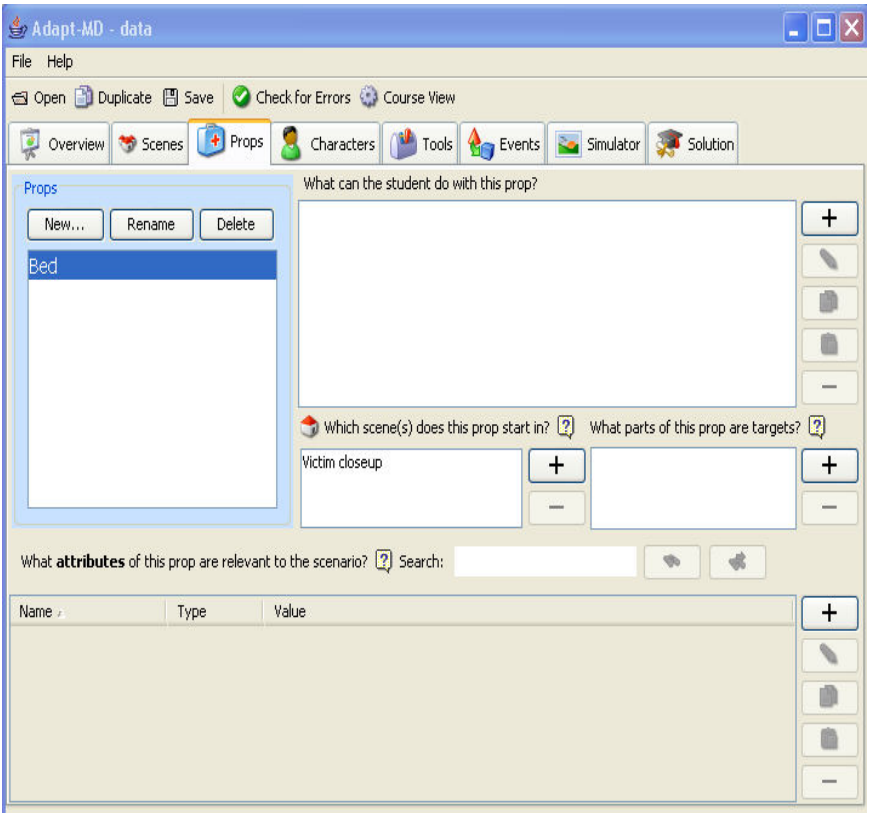

Fig. 2. SimCore Authoring Tool

Following the physical layout of the scenario build, the Characters are added to the scenario. Characters represent victims, bystanders and other people that are not players in the scenario. Locations, Props, Tools, and Characters can all have Actions associated with them. Actions either cause something to happen or report on the state of something in the scenario. For example, “check pulse” would be an Action that reports the state of a Character's pulse, whereas “bandage arm” would be an Action that causes something to happen.

Characters and Props have a series of attributes that allow the simulator to keep track of the state of the object during the simulation. The authoring tool includes domain specific wizards that help the author populate a number of default attributes (Figure 3). The types of data collected from the wizards were based on feedback from experienced EMS trainers.

Events are series of Actions that are not tied to Locations, Props, Tools or Characters. Events trigger on timers and other world state changes and execute their Actions without a player's direct influence. In addition, an event can end up triggering another event, allowing the author to create event chains for specific triggers.

Eventually, it is necessary to link the scenario to a simulator of some sort. The standard version of the SimCore authoring tool includes a way to link the scenario to a Flash based simulator. This allows the author to add pictures and connect these pictures to Locations, Props and Characters in the scenario. If the scenario is being developed for another simulator GUI, then there will be a tab for providing the same sort of links between the scenario and the simulator GUI.

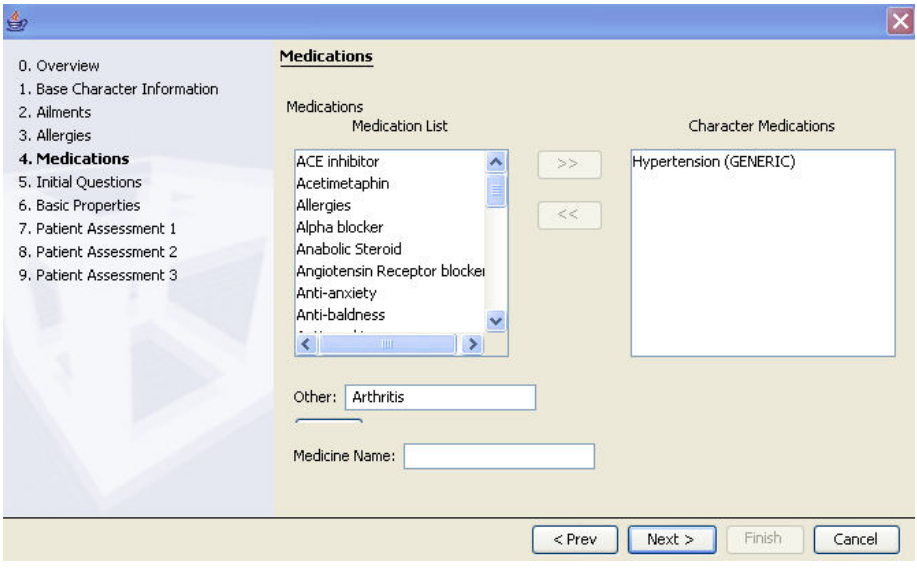

Fig. 3. Adding Medications for a Character in the Authoring Tool

Finally, there is a section in the authoring tool for outlining the performance assessment criteria expressed as a specification of the correct steps the student player is supposed to perform in the scenario. A solution template is a tree made of nested nodes which can be simple task nodes or group task nodes. Each simple task node recognizes a single student action. It specifies:

- An action pattern that can be matched by certain actions,
- An optional simulation state condition that specifies a constraint on the values of one or more simulation state variables that must be satisfied in order for the task node's action to be appropriate at a given time,
- zero or more principles that are demonstrated when the student carries out a matching action when the simulation state condition is satisfied,
- optional text that is displayed as a hint to the student, and
- optional text explaining the justification for the action. This text is shown during after-action review to explain the significance of a particular action.

Each group task node contains one or more simple task nodes and/or lower level group task nodes. It has the following information associated with it:

- An optional description that serves mostly as annotation
- A condition field whose intent changes depending on whether the node has been marked mandatory or conditional. Mandatory nodes have an associated pre-condition field that specifies the condition that must be satisfied before the actions in the group can be performed. Nodes marked Conditional have a condition field that states when the node is valid.
- Zero or more learning objectives that are demonstrated when the student carries out the actions that are recognized by the simple and (sub) group task nodes in this group.
- Justification that explains the reason for the task.

Each group task node can be configured so that it is satisfied when the student carries out actions that match (or satisfy) all of its children nodes, or just one of the children nodes.

In addition, there is a mechanism to incorporate expected errors as specified by the author. Expert instructors are often aware of the typical mistakes made by novice learners. With error rules, SimCore provides scenario authors with a way to recognize these types of mistakes and provide specific feedback to the student. For example, an instructor may know from experience that novice paramedics typically forget to wear gloves before administering fluids. An error rule can be used to recognize this and provide feedback on the importance of wearing gloves to avoid possible infections.

Associated with each error rule is text feedback that will be sent to the simulator when an error rule is matched by a student action. Similarly, authors can specify explanations with error rules that are sent to the simulator as a part of the after-action review. This is intended as a detailed contextual explanation of the error with enough information to help the student avoid this error in the future.

6 Evaluation

The SimCore framework was developed in three spirals. The software at the end of each spiral was demonstrated to experienced EMS trainers. Each time, the expert suggestions have strongly influenced the design for the next spiral and for the final system. We are currently making the product available to beta testers and we will be collecting feedback from these users.

7 Related Work

A few intelligent tutoring systems have been developed for healthcare domains [8,9]. In most such systems, the tutoring is tightly coupled with the simulation or the problem-solving interface. SimCore is based on a domain-independent scenario model and can thus be used for a broad range of domains. It is designed to be plugged into third-party simulators with minimal effort. Murray [7] presents an in-depth discussion of authoring tools for Intelligent Tutoring Systems. The trade-off between power and usability is a central dimension that characterizes authoring tools. Our objective is to develop a tool that is usable by healthcare training experts who have

little programming experience. Hence our first version of SimCore trades power for authoring simplicity. Future versions of the system will make the framework increasingly powerful while involving users in the development process to ensure that usability does not get sacrificed.

8 Future Work

SimCore beta is now available. SimCore is re-usable framework for developing low-cost, customizable simulations and training games for training EMS and other healthcare professionals. Using SimCore, subject matter experts and course developers can rapidly create simulations to test students' ability to apply their skills and decision-making capability in realistic scenarios. SimCore provides the facility to incorporate automated performance assessment and feedback, enabling organizations to create rapidly powerful, skill-application oriented self-paced training. A Flash simulator that is included as a part of the SimCore package makes the system usable by organizations that do not have a third-party simulator at their disposal.

In the near future, our primary objective to collect feedback from beta users on the system, understand how well it meets their needs, and gaps that must be addressed by future versions. Efficiency related improvements are also in the near-term agenda.

Our long-term vision for the product includes improved usability, enhanced student modeling, adaptive coaching, and reflective after-action review with Socratic dialogs.

References

1. Schank, R.: What We Learn When We Learn by Doing, Technical Report no. 60, Institute of Learning Sciences, Illinois (1995)
2. Johnson, W.B., Norton, J.E.: Modeling student performance in diagnostic tasks: A decade of evolution. In: Regian, J.W, Shute, V.J. (eds.) Cognitive Approaches to Automated Instruction, Lawrence Erlbaum Associates, Mahwah (1992)
3. Madni, A.M., Madni, C.C., Sorensen, H.B., Garcia, S.K.: Intelligent agents for individual and team training applications. In: IEEE Proceedings of the Man, Systems and Cybernetics Conference (October 2005)
4. Sorensen, H.B., Babbitt, Bettina,: Training medical professionals using interactive simulation. In: Research on Instructional Competencies, AACE, World Conference on Educational Multimedia, Hypermedia and Telecommunications (June 2003)
5. Kolb, D.A.: Experiential Learning. Prentice-Hall, Englewood Cliffs, NJ (1984)
6. Forbus, K.D., Feltovich, P.J.: Smart Machines in Education. AAAI Press, California (2001)
7. Murray, T., Blessing, S., Ainsworth, S.: Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive, and Intelligent Educational Software. Springer, Heidelberg (2003)
8. Shaw, E., Ganeshan, R., Johnson, W.L., Millar, D.: Building a Case for Agent-Assisted Learning as a Catalyst for Curriculum Reform in Medical Education. In: Proceedings of the Int'l Conf. on Artificial Intelligence in Education (July 1999)
9. Kizakevich, P.N., McCartney, M.L., Nissman, D.B., Starko, K., Ty Smith, N.: Virtual Medical Trainer: Patient Assessment and Trauma Care Simulator. In: Westwood, J.D., Hoffman, H.M., Stredney, D., Weghorst, S.J. (eds.) Medicine Meets Virtual Reality - Art, Science, Technology: Healthcare (R)evolution, pp. 309–315. IOS Press and Ohmsha, Amsterdam (1998)