Simulation Development and Authoring: Why Abstraction Matters

Dr. Sowmya Ramachandran, Randy Jensen, Erik Sincoff
Stottler Henke Associates, Inc.
San Mateo, CA
Sowmya@stottlerhenke.com, Jensen@stottlerhenke.com, Sincoff@stottlerhenke.com

ABSTRACT

The use of simulations for training has achieved mainstream status. While they span a range from high-fidelity simulators with specialized hardware to those with simple branching interactions, the latter are fast gaining in popularity given their capacity to provide experiential learning at a relatively low cost. As they become commonplace, there will be a drive towards increasing simulation complexity. To date, many of these branching simulations are implemented using hyperlinks and other content navigation mechanisms. Very little state information is maintained explicitly, if any. Instead, state information is embedded within the navigational structure. Although this is adequate for simple simulations and has the advantage of intuitive authoring, this representation does not scale easily as the need for complexity rises. An alternative is a representation where simulation states are explicitly maintained and used to drive the simulation. This is employed by all high-end simulations. The challenge is to provide a way to create state-based simulations development platform that is also intuitive in terms of authoring, and does not increase the cost of content significantly. In this paper, we will describe an approach that facilitates state-based simulations while also providing ease of authoring. We will also discuss the costs and benefits of both state-based and alternate representations.

ABOUT THE AUTHORS

Dr. Sowmya Ramachandran is a research scientist at Stottler Henke Associates. Her work focuses on the applications of AI to improve education and training. She leads research and development efforts on Intelligent Tutoring Systems and Authoring Tools. She has developed intelligent tutors for a diverse range of military and civilian domains. Dr. Ramachandran headed the development of ReadInsight, a tutor for teaching reading comprehension skills to adult English speakers. She also led the development of a tutor for training Tactical Action Officers in the Navy. This system uses natural language processing technologies to assess and train TAOs and is currently in operational use at the Surface Warfare Officers School. Currently, she is leading an investigation of techniques for developing adaptive expertise in trainees through the use of intelligent tutoring, and for developing automated analysis of chat-based communications to support after-action review for large team training exercises. Dr. Ramachandran has presented extensively at conferences and has participated in a number of panels for defining roadmaps for future Intelligent Tutoring Systems research.

Randy Jensen is a group manager at Stottler Henke Associates, Inc., working in training systems since 1993. He has developed numerous Intelligent Tutoring Systems for Stottler Henke, as well as authoring tools, simulation controls, after action review tools, and natural language analysis methods. He is currently leading projects to develop an embedded training Intelligent Tutor for the Army, and an authoring tool for virtual training demonstrations for the Army. He holds a B.S. with honors in symbolic systems from Stanford University.

Erik Sincoff is a lead software engineer at Stottler Henke Associates, and has focused on the design and implementation of intelligent tutoring systems in a variety of domains including emergency medical training, military equipment training, and homeland security training. Previously, he was a senior software engineer at Teknowledge Corporation, where he worked in numerous domains, including implementing tutors in multiuser worlds. He has been at Stottler Henke since 2005 and has a MS in computer science from Stanford University.

Simulation Development and Authoring: Why Abstraction Matters

Dr. Sowmya Ramachandran, Randy Jensen, Erik Sincoff
Stottler Henke Associates, Inc.
San Mateo, CA
Sowmya@stottlerhenke.com, Jensen@stottlerhenke.com, Sincoff@stottlerhenke.com

INTRODUCTION

The use of simulations for training has achieved mainstream status. While they span a range from high-fidelity simulators with specialized hardware to those with simple branching interactions, the latter are fast gaining in popularity given their capacity for providing experiential learning at a relatively low cost. As they become commonplace, there will be a drive towards increasing simulation complexity. To date, many of these branching simulations are implemented using hyperlinks and other content navigation mechanisms. In such cases, very little state information is maintained explicitly, if any. Most of this information is embedded within the navigational framework. Essentially the context within the simulation provides the implicit state.

Although this is adequate for simple simulations and has the advantage of intuitive authoring in such cases, this representation does not scale easily as the need for complexity rises. Frequently, training or analysis requirements make it necessary to construct assessment mechanisms that monitor the simulation and make determinations about performance. In a highly contextualized approach, the dependence on scenario context for implicit state information results in a requirement that scenarios and simulation sequencing enforce the presumptions in the assessment mechanisms. This in turn makes the authoring complexity increase sharply with more complex simulations, rapidly approaching levels that are no longer practical without explicit states.

An alternative is a representation where simulation states are explicitly maintained and drive the simulation. This is employed by all high-end simulations as a direct consequence of the range of complex scenarios that they attempt to model. This can be considered an abstracted approach in the sense that states are abstracted from context. Using a simple procedural example, consider a simulation for mechanical maintenance on a vehicle. For training purposes, the simulation may be used to evaluate performance on a well-defined procedure, and so the notion of states primarily corresponds to completed tasks or procedures (e.g., checked fluid level, disconnected battery, etc.). In an abstracted state-based approach, the user may be allowed to work in any context in the vehicle at any time. Performance assessment is a matter of looking for the satisfaction of prerequisite states before tasks are completed. Using this general abstracted approach, scenario context may be easily introduced into assessment logic when needed, without being a necessary factor at all stages. In contrast, in a contextualized approach with implicit states, it becomes necessary to either structure the simulation and scenarios to restrict the user's actions, or introduce assessment logic that can interpret a combinatorial explosion of possible user sequences. Barring either of these remedies, it would be impossible for assessment mechanisms to determine the correctness of an action in a particular context, because of the uncertainty about whether the proper prerequisite tasks have been completed.

Although a state-based approach is nearly inescapable for high-end complex simulations, it is also relevant at the opposite end of the spectrum, where lightweight simulations are increasingly being used for experiential training in lower fidelity arenas. The challenge is to provide a way to create a state-based simulation development platform that is also intuitive in terms of authoring, and does not increase the cost of content significantly. In this paper, we will describe an approach that facilitates state-based simulations while also providing ease of authoring. We will also discuss the costs and benefits of both state-based and alternate representations.

BACKGROUND

The role of simulations in military training has been well established in recent years. This paper focuses on training simulations used as tools for the conduct of scenario-based training (SBT) or event-based approach training (EBAT) (Prince, Oser, Salas, & Woodruff, 1993; Oser, Cannon-Bowers, Salas, & Dwyer, 1999). SBT and EBAT refer to a similar concept of providing

a training audience the opportunity to practice the application of knowledge, skills and attitudes (KSAs) and receive feedback on their performance. This fundamentally differs from lecture-based training in the sense that "the scenarios become the curriculum – meaning the events inserted in the scenarios constitute the learning objectives, the means to achieve the desired learning outcomes" (Salas, Priest, Wilson, & Burke, 2006). The implication is that carefully defined training objectives must be used to construct scenarios that require trainees to demonstrate competence in situations of varying difficulty and timing (Stretton & Johnston, 1997).

In order to deliver meaningful feedback that trainees can use to modify their models and future behavior, scenarios must support performance assessment based on either automated measures or human observations. Scenarios constructed with explicit friction points or triggers inserted afford human observers or automated evaluation mechanisms a concrete basis for measuring performance. Salas, Priest, Wilson, & Burke (2006) elaborate on the importance of multi-dimensional measurement, which "should be tied to learning objectives and scenario events to ensure the assessment of whether or not targeted competencies are learned (i.e., outcome) and why performance occurred as it did (i.e., process)."

This underscores the key objectives in constructing a training simulation to be used for scenario-based training. It must provide the opportunities for practice in an environment with sufficient fidelity to create realistic demands on trainees while performing tasks or decisions. Performance must be measurable in the simulation, and particularly where automated assessment methods will be implemented with the simulation, it is important to be able to disambiguate the conditions under which potentially erroneous actions are made, in order to generate effective trainee feedback.

The matter of using empirical evidence to determine the optimal level of fidelity for a training simulation in a given domain remains a challenging task. However, as an incremental step, researchers have arrived at a commonly accepted breakdown of three dimensions: physical fidelity, functional fidelity, and psychological fidelity (Hays & Singer, 1989). Physical fidelity typically is used to refer to "look and feel" correspondence between a simulation environment and the real world. Functional fidelity is sometimes treated as a subcategory of physical fidelity, in that it concerns the nature of tasks performed in the simulated environment, and the similarity to how such operations take place in real situations. Cognitive or psychological fidelity refers to the ability of a simulation to create the conditions where a trainee goes through the same cognitive processes as the real world operational environment requires for problem-solving, decision-making, and so on.

Despite the visceral appeal of high fidelity simulation, researchers have produced evidence that simulations lower in physical and/or functional fidelity can still yield training results that are effective and in some cases comparable (Wickens & Hollands, 2000). Estock, Steltzer, Alexander, & Engel (2009) reported on side-by-side experiments conducted with subjects using two F-16 aircraft simulators with differing levels of "cockpit fidelity." The study found that although subjects' subjective assessments rated the lower fidelity simulator lower for overall effectiveness, objective measures of training effectiveness showed no differences. This result and similar results from other studies may be important factors to consider in simulation development, particularly where practical matters of cost and complexity may be primary factors. As might be expected, several researchers have also identified how task-related variables can impact the role that different forms of fidelity play in effectively training certain learning objectives (Hays & Singer, 1989). For example, consider a planned training system that aims to exercise trainees with targeting decisions, as opposed to visual target acquisition skills. In such a case, it is less important to construct a high fidelity simulation environment providing the visual cues to identify targets, when the primary learning objective concerns trainee decisions and actions *given* the presence of known and designated targets.

Thus, simulation authoring may take a low fidelity approach for either practical reasons or direct awareness of the limited requirements presented by the tasks to be modeled, as long as sufficient psychological fidelity can be preserved for such domains. This is essentially the context for this paper's discussion of the value of an abstracted state-based approach in low fidelity simulations. Although many of the same premises of a state-based approach apply to high fidelity simulations as well, the modeling complexity of these simulations nearly always requires a state-based paradigm. However, with lower fidelity branching simulations, the near term expedient of a contextualized approach often leads to a limitation for scalability needs later on.

Lightweight simulations are often constructed for procedural domains, structured with a simple branching approach that generally resembles the prescribed flow

of actions for the operational domain. However, in practice, the performance of procedural skills often involves mixes of ordered and un-ordered procedures. If these are modeled with implicit states, where all performance is considered in terms of the ordering of trainee actions, then this can make the authoring process extremely complex because of the possible sequencing permutations. By introducing states, performance measures can be more general, and look not only for conditions where tasks preceded tasks, but also where states preceded tasks.

Explicit states also make it easier to construct training that targets distinct learning objectives, and measures progress toward those objectives independently. When states are abstracted from simulation contexts, any contextual point in the training sequence can be presented without requiring the subject to go through a prescribed navigational path to establish an implicit state. This means an author or instructor can more easily re-purpose a fragment of a scenario for targeted practice. Whether such focused practice is for remediation, refresher training, or simply tailoring to the weaknesses of an individual learner, this is particularly valuable when building adaptive instruction.

The following sections will present a state-based representation for a simulation/scenario and discuss how it compares with other representations. Our discussions will refer tothe area of Emergency Medical Services (EMS), specifically simulations for training paramedics. In particular, we will use as a running example a scenario that involves a child who has been run over by a runaway car in his driveway and is suffering trauma. It opens with the player surveying the driveway that contains the car, a man sitting by the car, and a child crying on a woman's lap. The player is required to determine who is hurt and take appropriate actions and administer treatment

AN APPROACH TO STATE-BASED SIMULATIONS

Simulation Model

The state of the simulation, conceptually, should represent information about its meaningful and manipulable components. A simulation typically represents a circumscribed perspective on the real-world. It is reasonable then to maintain state information that reflects this perspective. The ontology of the model will vary depending on the purpose of the simulation (e.g. for troubleshooting vs for training people in making sales calls).

For the purposes of our discussion we will limit our attention to procedural simulations such as the one for training paramedics. We model a simulation as a set of locations, props, Non-Playing Characters (NPCs), 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 child hurt in the accident, as well as the other people on the scene would be modeled as NPCs.

Each entity in the scenario (i.e. each location, prop, NPC) has a set of associated attributes. The total set of attributes essentially defines the simulation state. For example, a car might have attributes representing the state of its parking brake (“engaged” or “not engaged”), an NPC representing an accident victim would have a large set of attributes to describe his/her physiological state, and an ambulance might have attributes describing the number of beds available, and its current location. As the simulation unfolds, these attributes will be updated to reflect the current state. If the learner decides to engage the car’s parking brake, for instance, its attribute will be changed to reflect that it is now parked. This information is always available to the simulation, making it possible, for instance, to include a simulation event where the car rolls down the driveway and onto the street if the user fails to engage its parking brake within an expected window of time. This enables simulation developers to introduce training events designed to address specific learning objectives (Section “Background”).

The scenario model includes actions that can be performed on the NPCs and props. Each action has a unique command name, 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. The “Engage Parking Brake” action mentioned above has no preconditions and produces the effect of setting the state of the car’s parking brake to “engaged”. For example, “Check Blood Pressure” may be an action with the precondition that the blood pressure monitor should be available for performing this action, and the effect of reporting back the blood pressure of the NPC in question. The action definition would also include rules specifying the simulator's response when the player performs this action. The scenario model

additionally would include events which are a set of simulation-initiated effects that will be performed when a set of preconditions are met. Events make it possible to create dynamic simulations where the state of the world can change even in the absence of any inputs from the student, thus enabling the embedding of situations that call upon skills. We described one such event above where the car rolls onto the street in the simulation when the student fails to engage its parking brake, thus emphasizing the point that securing scene safety is an important goal for a paramedic while attending to an emergency. Actions and events are the means for effecting changes to simulation state. Thus, the simulation starts out in an initial state as specified by authors and evolves dynamically as dictated by actions and events. It is important to note that while the state information is affected by actions and events, it is maintained separately from them.

Notice that the simulation model does not include any reference to the user interface or particular interaction elements. The model can exist independently of its external visualization. This has the benefit that the external look and feel of a simulation can be changed without touching any of its internals. Similarly, the internals of the simulation can be changed without modifying the interface. The two can be developed separately and linked up by mapping model elements to interface elements.

ASSESSMENT UNDER A STATE-BASED REPRESENTATION

Typically, simulations (on the low-fidelity end of the spectrum) tie evaluations of learners with user interaction elements. For instance, a simulation might present the learner with a few treatment options with instructions to choose one. Each option might lead to a different path through the scenario, but ultimately one of the choices will provide positive or negative feedback. Here assessment is closely integrated with the interface. Changing the assessment rules entails changing all the relevant interface structures. Embedding the assessment logic with the navigation structure in this manner can cause authoring effort to increase rapidly with the desired sophistication of assessment.

A state-based representation frees assessment and simulation context from the interface. We can now define assessment rules that look for simulation state changes to evaluate student actions. For example, if the expected protocol states that the learner must check a set of vital signs before proceeding to treatment, but they may be checked in any order, the simulation can wait until the student starts treatments and then verify with the simulation state that all the requisite checks have been performed (the patient's attribute set will include flags to indicate if these actions have been performed).

To illustrate, the simulation model may include a patient NPC called *Patient1*. Its attribute set would include the following flags: *PulseChecked*, *BloodPressureChecked*, *CirculationChecked*, *PulseOximetryChecked*, *BreathingChecked* etc. In addition, the simulation will have the following assessment rule to score the student's performance:

> If the current action is a treatment action, then
>
> Set *VitalSignsScore* to *1*, if all (*Pulsechecked*, *BloodPressureChecked*, *CirculationChecked*, *PulseOximetryChecked*, *BreathingChecked*) are all *true*
>
> Set *VitalSignScore* to *0* otherwise.

Each of the actions of checking pulse, blood pressure, circulation, pulse oximetry, and breathing will update the corresponding state variable. Subsequently, an assessment module will refer to these variables rather than the actions themselves. With rules of this kind, we can implement assessment criteria that disregard the order in which actions are performed. To accomplish this without a state representation would involve developing a separate navigation path to represent each specific ordering of actions. This is clearly not desirable.

SIMULATION AUTHORING

Authoring complexity is a crucial consideration in the design of simulation development tools. A factor contributing to the increasing popularity of simulations has been that user-friendly authoring tools have been placed in the hands of content experts. To keep simulations viable, the locus of authoring should continue to reside with the content experts. However, this impacts the degree of authoring complexity that can be supported without making it impossibly burdensome. We have argued that branching simulations are limited in the complexity they support. The state-based approach facilitates more complex interactions and assessments but brings with it the cost

of increased authoring effort. It is important to design authoring tools for state-based simulations that mitigate this effect. Here we describe a tool for authoring content for the simulation model described in the previous section. Building a scenario can be broken down into the following parts: (1) designing the physical world, (2) populating the world, (3) adding events to the world, (4) mapping the scenario world to the simulator interface, and (5) keeping track of what the student is doing. There are form-based editors for each of these steps. Figure 1 shows the main authoring interface. Note the tabs, one for each simulation model component. Authoring to a large extent involves creating simulation entities (i.e. Props and NPCs) and filling out their attributes. This can be further facilitated by the inclusion of domain-specific wizards to create and initialize attributes. Figure 2 shows a wizard for the paramedic simulation. Figure 3 shows the form for defining an action.

Authoring actions, events and simulation rules requires more effort and skill not only for specifying the parameters, but also for conceptualizing and modeling the simulation prior to authoring. Modeling is an additional step that must be included in the simulation creation process. The authoring tool provides form-based editors and wizards to facilitate the actual specification of actions, events and rules.

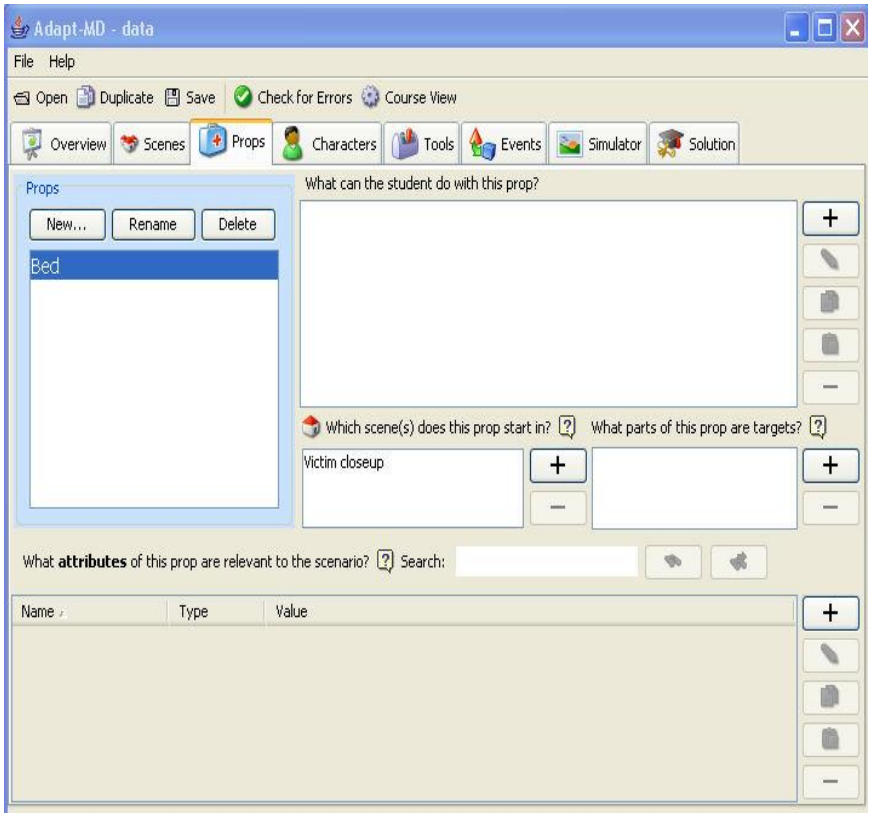

Figure 1. Authoring Interface for a State-Based Representation

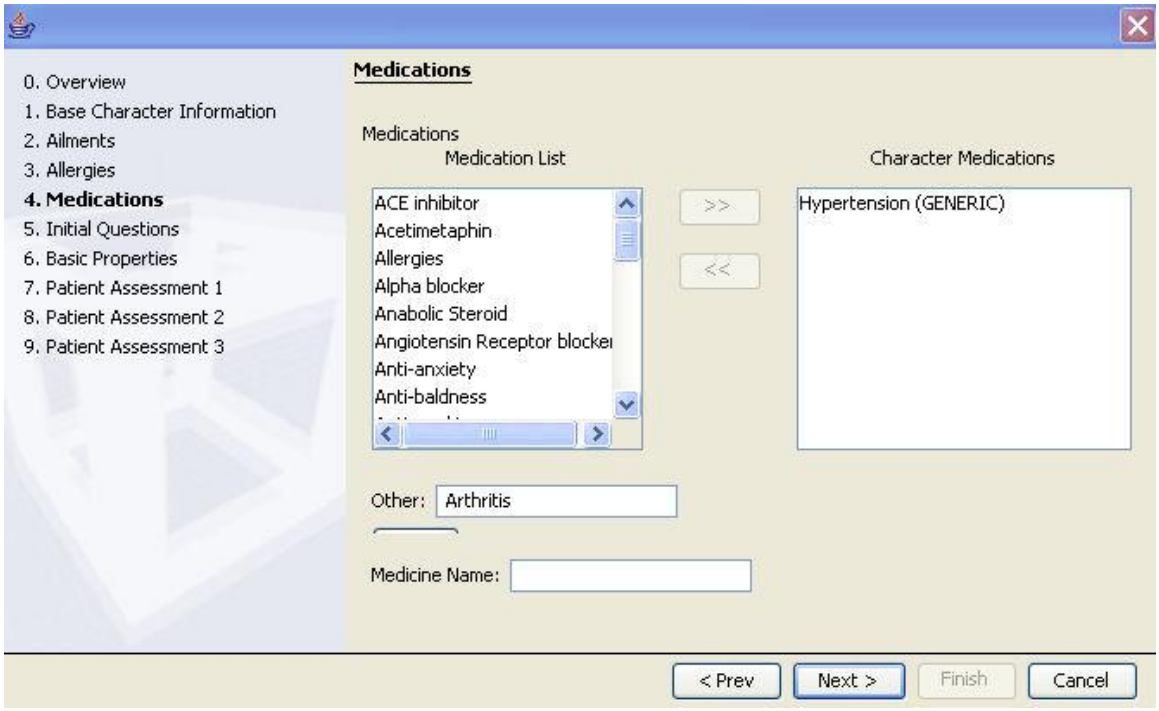

Figure 2. Adding Medications for a Character with the Authoring Tool

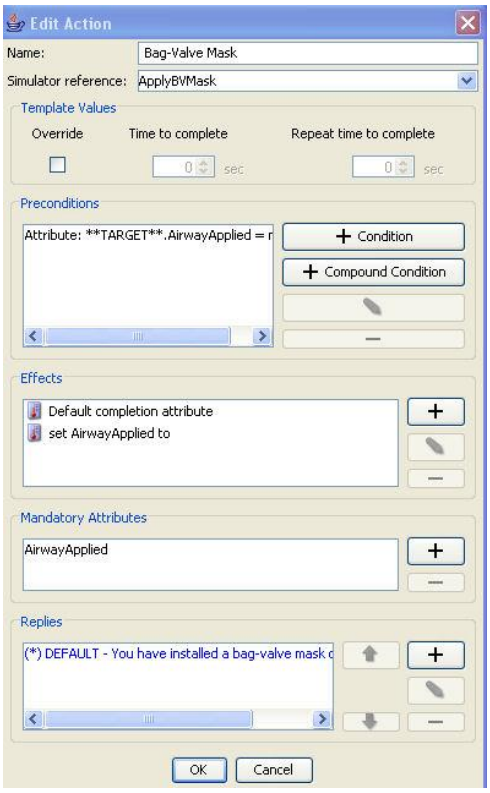

Figure 3. Authoring Form for Defining a New Action

ANALYSIS

An easy hyperlink-based approach to implementing our EMS scenario is to ask the trainee a series of multiple-choice questions to simulate their assessment of the patient's condition. While this is easy to author, it will result in a scripted scenario and take most of the initiative away from the student. Students will have limited freedom to determine the sequence of tests to perform and treatments to deliver. Allowing the student more degrees of freedom will make the branching logic rapidly unmanageable.

There are many commercial off-the-shelf simulation development tools that go beyond the type of branching simulation described above (e.g. the Lectora™ autoring suite by Trivantis and the Toolbook™ authoring suite by SumTotal Systems). These tools include a notion of state by allowing variables to be associated with simulation interface elements. Interactions can be in the form of menus and hotspots. Simulations can be developed using these tools that give the learner the degrees of freedom required to elicit his/her initiative in determining the right sequence of actions. Scenarios do not have to be tightly scripted and can thus provide a more realistic assessment of the learners' skills. Tying the variables to the interaction elements, however, has disadvantages. Suppose the paramedic scenario requires the student to transport the patients in an ambulance while administering treatments. This will typically require a different location and a different set of interactions. Consequently the state information associated with the earlier context is lost. This is at times undesirable. Patients' vital signs, the set of treatments administered, to name a few state variables, are all still relevant and salient in the new context and should be carried over.

As a workaround, many of these simulation development tools allow global variables, i.e. variables that are not associated with any particular interface element and are available independent of context. For the paramedic example scenario, one could create global variables that represent all the required entities' attributes. Thus, authors can define variables to represent the child's pulse, breathing, blood pressure etc. However, the separation between state and interface is not as robust or clean as with the approach we have described. Furthermore, they do not provide a way to model the entities themselves explicitly. This leads to some limitations. For example, the model that we have described makes is possible to define an action called Check Blood Pressure on a general "person" NPC and apply that to different simulated people (e.g. Check Blood Pressure applied to child, Check Blood Pressure applied to the other man in the scene). This leads to efficiencies in authoring and representation that is not possible otherwise. Additionally, our model provides the benefit of allowing state information to be tied to conceptual scenario entities and thus to be carried over regardless of the interaction context. Whenever an entity is relevant to a context, its state information is readily available.

Authoring complexity is an important consideration when it comes to simulation development tools. Authoring simulations with branching hyperlinks is on the easy end of the spectrum. Introducing variables and scripts increases the complexity of authoring and design. The introduction of abstractions and programming constructs makes authoring less accessible to those that are not trained as computer programmers. However, unlike the high-fidelity simulators, authoring these simulations does not have to be done at the actual programming level. The authoring tools only require managing variables and writing simple events and action logic.

The demand for more interactive and effective training is driving the quest for better simulations at lower costs. The current popular methods of simulation development will hit soon hit a wall unless they evolve to meet the growing needs for interactivity and sophistication. The good news that many of the COTS simulation authoring tools are indeed evolving in the direction of state-based representations such as the one discussed here. However, we argue the envelope of abstraction can be pushed further in a direction that will significantly expand the possibilities for the types of simulations that can be developed while keeping authoring complexity down at a level that is suitable for content experts.

REFERENCES

Estock, J. L., Stelzer, E. M., Alexander, A. L., Engel, K. (2009). Is cockpit fidelity important for effective training? Perception versus performance. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC 2009).*

Hays, R. T., & Singer, M. J. (1989). *Simulation Fidelity in Training System Design: Bridging the Gap Between Reality and Training.* Springer-Verlag, New York, NY..

Oser, R. L., Cannon-Bowers, J. A., Salas, E., & Dwyer, D. J. (1999). Enhancing human performance in

technology-rich environments: Guidelines for scenario-based training. In E. Salas (Ed.), *Human/technology interaction in complex systems* (pp. 175-202). Stamford, CT: JAI Press.

Prince, C., Oser, R., Salas, E., & Woodruff, W. (1993). Increasing hits and reducing misses in CRM/LOS scenarios: Guidelines for simulator scenario development. *International Journal of Aviation Psychology, 3*(1), 69-82.

Salas, E., Priest, H. A., Wilson, K. A., & Burke, C. S. (2006). Scenario-based training: Improving military mission performance and adaptability. In C. A. C. A.B. Adler, and T.W. Britt (Eds.), *Military life: The psychology of serving in peace and combat* (Vol. 2: Operational Stress, pp. 32-53). Westport, CT: Praeger Security International.

Wickens, C. D. & Hollands, J. G. (2000). *Engineering psychology and human performance: Third edition.* Upper Saddle River, NJ: Prentice Hall.

Stretton, M. L., & Johnston, J. H., (1997). Scenario – based training: An architecture for intelligent even selection. In *Proceedings of the 19th Annual Meeting of the Interservice/Industry Training Systems Conference* (pp. 108-117). Washington, DC: National Training Systems Association.