Authoring Tools for Free-Play Simulation-Based Troubleshooting Training

Eric Domeshek, Sowmya Ramachandran, Randy Jensen
Stottler Henke Associates, Inc
San Mateo, CA
domeshek@shai.com, sowmya@shai.com, jensen@shai.com

ABSTRACT

All training systems require definition of curriculum and creation of associated instructional materials. Simulation-based training systems further require development (or at least integration) of a suitable simulation environment, and specification of challenge problems. Intelligent tutoring systems (ITSs), in addition, require logic for automated assessment, student/tutor interaction, and experience sequencing. Costs to create all these pieces can mount quickly, so it is important to provide authoring tools tied to general methods and code to speed content creation, reduce required levels of authoring expertise, and facilitate reuse.

This paper reports on authoring tool development and preliminary evaluation in the context of a simulation-based ITS employing novel mechanisms for tutoring troubleshooting skills. Major challenges included: (1) defining the data needs of the ITS, given it is neither a rule-driven cognitive tutor, nor an example tracing tutor; (2) mapping the system's complex data structures to consistent, supportive, tailorable user interface conventions so as to enhance tool usability and maintainability; (3) bridging between the desired unified authoring tool suite and useful commercial off-the-shelf (COTS) tools—e.g., for didactic instruction and simulation; and (4) designing for flexibility, so as to track evolution of the underlying tutor mechanisms and enable future generalization.

The example ITS trains Navy Information Systems Technicians to carry out troubleshooting and procedure-following tasks using virtualized computer networks as a free-play simulation environment. The tutor adds simulation instrumentation to gather information that feeds models of available actions and resulting states of knowledge. It then monitors student activity, hypotheses, and inferences, so as to provide contextualized, adaptive feedback and coaching. The project was concerned with establishing a general framework for simulation-based technology troubleshooting applications, and had strong requirements for broad-spectrum authoring tools. We describe the project context, tutoring approach, authoring tools developed, authoring evaluation feedback, relationship to other authoring tool efforts, and lessons learned.

ABOUT THE AUTHORS

Dr. Eric Domeshek is an AI project manager at Stottler Henke Associates, Inc. where he leads and supports projects applying AI technology to problems in training and decision support. He has worked on a wide range of ITSs and related training, education, and simulation environments spanning applications to military tactics, medical diagnosis, engineering systems management, business decision-making, and historical analysis. He is particularly interested in exploration of Socratic tutoring techniques and the development of authoring tools. He led the work on the authoring tools for a novel simulation-based technology troubleshooting ITS recently developed for the U.S. Navy. Dr. Domeshek received his Ph.D. in Computer Science from Yale University, focused on case-based reasoning. For his dissertation, he developed representations of decision rationale for social situations, intended to support case retrieval; this included extensive representations of characters' relationships, traits, and motivational structures. He served as research faculty at the Georgia Institute of Technology College of Computing where he contributed to the development of a line of case-based design aids. He was also an assistant professor at Northwestern University, developing goal-based scenario training systems at the Institute for the Learning Sciences.

Dr. Sowmya Ramachandran is a research scientist at Stottler Henke Associates where her research focuses on the application of Artificial Intelligence (AI) and Machine Learning to improve education and training. She leads research and development of ITSs and ITS authoring tools for a diverse range of military and civilian domains. Dr.

Ramachandran headed the development of ReadInsight, an intelligent 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. Most recently she led the development of an ITS for training U.S. Navy Information Systems Technicians in troubleshooting and maintenance skills. Dr. Ramachandran holds a Ph.D. from The University of Texas at Austin. For her dissertation, she developed a novel machine learning technique for constructing Bayesian Network models from data.

Mr. Randy Jensen is a group manager at Stottler Henke Associates, Inc., working in training systems since 1993. His research areas include adaptive training, distributed learning, game-based training, behavior modeling, and natural language processing. He has led projects to develop Intelligent Tutoring Systems and automated after action review tools for the Army, Air Force, Navy, and Marines. Recent work includes a model-based performance assessment capability for training troubleshooting skills in an intelligent tutor for the U.S. Navy. He also recently led the development of a game-based trainer for small unit tactical decision-making at the U.S. Military Academy at West Point. Mr. Jensen holds a B.S. with honors in symbolic systems from Stanford University.

Authoring Tools for Free-Play Simulation-Based Troubleshooting Training

Eric Domeshek, Sowmya Ramachandran, Randy Jensen
Stottler Henke Associates, Inc
San Mateo, CA
domeshek@stottlerhenke.com, sowmya@stottlerhenke.com, jensen@stottlerhenke.com

INTRODUCTION

There is good reason to pursue simulation-based training (Salas, Milham, & Bowers, 2003; Smith, 2014). Likewise, there are proven advantages to providing one-on-one instruction (Bloom, 1984). Given the cost and availability of human instructors, however, it is often effective to provide such instruction using intelligent tutoring systems (ITS; VanLehn, 2011; Ma et al., 2014; Kulik & Fletcher,2016). Unfortunately, the potential scope of application of such trainers is impacted by high content development costs associated with simulation-based ITSs. The content required for simulation-based ITSs includes at least the following (and often other elements as well):

- Curriculum
- Instructional Materials
- Simulation Environment
- Challenge Problems
- Automated Assessments
- Student/Tutor Interactions
- Experience Sequencing Constraints

This paper discusses an approach to designing, building, and vetting a comprehensive authoring tool suite for a large-scale military simulation-based ITS. Among the factors that make this work distinctive are the focus on providing automated assessment and tutoring for a class of complex skills in the context of a free-play simulation of a quite complex environment. In particular, the authoring suite needs to support content preparation for a novel tutoring mechanism—neither a rule-driven cognitive tutor, nor an example tracing tutor—tuned to train students in troubleshooting technical systems in a realistic environment. Our example system tutors novice U.S. Navy Information System Technicians (ITs) in Windows computer and Cisco network administration, primarily in the context of assigned service tickets to be resolved in a virtualized version of a typical ship network environment. However, the general approach can be applied to training in any technical troubleshooting domain, where (simulated) student actions and/or system states can be monitored, and where action results and state transitions can be inferred. For the Navy, this could include anything from troubleshooting nuclear submarines to diagnosing and treating patients.

TRAINING SYSTEM CONTEXT

The Intelligent Tutoring Authoring and Delivery System (ITADS) provides up to 10 days of instruction and coached practice covering aspects of the IT A-School's Windows and network administration segments. It also provides a capstone experience in which students are confronted with a mix of Windows and network problems and tasks. Finally, it provides an evaluation mode (for both segments, and for the capstone) in which tutoring is turned off and students must work exercises without ITS guidance; the ITS continues to carry out assessment, but only reports its results to the instructors. The student runtime is entirely web-based, as is the accompanying instructors' console. The authoring suite is currently a single-user desktop application, but support for multiple users and possible web delivery are potential future tasks.

Each exercise has several phases. The exercise proper is preceded by a block of recommended instruction intended to ensure the student has some preparation for the issues that will arise in the simulation. This instruction takes the form of didactic presentations—largely PowerPoint decks converted for web-browser presentation, supplemented with

animations, video, and multiple-choice learning checks. Most exercises begin with the presentation of a service ticket and then proceed to coached activity in a simulated shipboard environment. Following the exercise there is an after-action review, in which the tutor presents critiques of student performance.

The phases of mainstream (diagnostic) simulation exercises include information gathering, diagnosis, fix, and wrap-up. This structure was designed to largely follow and emphasize the Navy's formal troubleshooting process (which students are taught, but previously had little opportunity to practice). For (procedural) exercises, where service tickets specify a task rather than reporting a problem, the student effectively starts work in the "fix" phase.

In either case, the simulation provides each student his or her own shipboard IT network, in a state that reflects the existence of a problem or readiness for an assigned task. This is accomplished using virtual machines (VMs) and substantial server infrastructure. We established a baseline configuration for a fictional Navy ship with a representative set of virtualized workstations, servers, routers, etc. Though reduced in number, these VMs accurately reflect Navy configurations and standards. We also designed a web user interface (UI) providing students access to the full range of relevant resources, including not only the shipboard spaces and associated IT systems (including both physical views of the hardware and access to software user interfaces), but also supporting documentation, logs, and limited access to virtual colleagues.

With some limitations, the student is given free rein to move around the ship, access IT systems, and take any actions on accessed systems using normal graphical user interface (GUI) or command line interaction. The simulation UI and the virtualized systems are instrumented to report relevant student actions to the ITS. The ITS tracks those actions against its model of the ship network state and consequent action results—emphasizing effects on *potential student knowledge of the system state*. Like most ITSs, this system has an expert model for performance assessment and context-sensitive coaching. That model captures inferences that should follow from the results produced by possible student actions. During troubleshooting, it maintains an expert hypothesis space and uses that to critique student actions. Furthermore, the tutor UI offers the student ways express their own hypotheses, and so the tutor can also track and critique aspects of the student's reasoning.

TROUBLESHOOTING ITS AUTHORING REQUIREMENTS

Given the goals and design above, we can better characterize several kinds of content required to drive such an ITS:

- **Curriculum Model:** A hierarchically organized set of curriculum nodes provides a skeleton on which all other content can be hung. Instruction, exercise scenarios, assessments, and dialogs are all tied back to curriculum nodes, allowing the ITS to track evolving student mastery of course materials, and to choose appropriate materials and activities to advance that mastery (sequencing).
- **Didactic Instruction:** Web-delivered media-enhanced briefing decks must be built and then linked into the system—i.e., labeled with curriculum nodes and associated with exercise scenarios. Commercial off-the-shelf (COTS) tools provide good support for instruction/media preparation. We had to pick leading COTS tools, specify conventions for their use, and build means to link their products into the ITS.
- **Simulation Configuration:** Each simulation-based scenario requires that the virtual shipboard IT systems be initialized to realistically represent the hypothetical Navy ship in an appropriate state (e.g., to embody the reported problem for a service ticket). Again, there are COTS tools for creating, configuring, freezing, and restarting VMs, and sets of inter-related VMs. So system integration issues again focused on choosing leading COTS tools, define standards for their use, and link their products into the ITS.
- **Scenario Configuration:** In addition to the ship network simulation, there are other elements—notably including the tutor and simulated crew members—that must be configured to launch an exercise. The scenario must be linked to the curriculum and instructional materials. The scenario's service ticket must be defined. Other materials such as an introductory narrative, supporting paperwork, and summary of the key training points are needed. Simulated crew members must be primed to respond to a relevant set of student questions that might reasonably be asked during troubleshooting. Finally, the tutor must be told what problem exists in the simulated IT systems, and what initial evidence the student is given.
- **Expert Model:** To enable automated assessment, the *content* of the expert model, must be strongly tailored to meet the specific training needs of the curriculum and supporting scenarios. However, it is still general to the extent that many elements can be reused across scenarios. Furthermore, the *structure* of the expert model

is designed to be suitable for a much wider range of troubleshooting domains. The core ontology supporting the assessment model includes several constructs—such as Activities, Problems, and Findings—that each include substantial nested structured data. Details are reported in (Jensen et al., 2016).

- **Student/Tutor Dialog Scripts:** In addition to the simple question/answer interactions with simulated crew members, and some algorithm-driven interactions with the simulated tutor, it is useful for an ITS to support a more general notion of extended scriptable dialogs. Such dialogs can be used in support of simulation-based exercises—e.g., to provide an interactive after-action review of the student's exercise performance. In the target system, dialogs are tree-structured, with each tree node specifying a set of tutor prompts, expectations for student input, and potential tutor responses to those inputs. The tutor descends the tree and pursues nested sub-topics when the student does not initially give evidence of understanding the point addressed by the parent node. The tutor can also score students on their mastery of curriculum nodes associated with dialog nodes (see Domeshek, Holman, & Luperfoy, 2004 for details).
- **Tutoring Policy Configuration:** In a complex simulation-based ITS, there are many degrees of freedom—i.e., instructional decisions that cannot be resolved definitively based on theory—for which it is desirable to offer instructors control. The target authoring tool thus also supports specifying tutoring policies, such as where to set the break points for classifying a student as a novice or a master of a curriculum point, how to pick the scenario for the next exercise, when to offer didactic instruction, and so on.

Our project team recognized that it would be unreasonable to expect all potential authors to have exactly the same range of knowledge and competencies. Some authors might be knowledgeable in the domain and facile with PowerPoint (thus able to prepare didactic content) but unfamiliar with VMs and VMWare (thus, unable configure and manage sets of VMs). Other authors might have familiarity with use of scripting languages and so might be better equipped to deal with aspects of the domain modeling and dialog scripting tasks. Accordingly, system requirements reflect some division of labor and corresponding definition of different classes of authors, provided access to different subsets of the overall authoring tool suite. This notion of "tiered" authoring tools was discussed in (Domeshek, Jensen, & Ramachandran, 2015).

TROUBLESHOOTING ITS AUTHORING TOOL DESIGN

Given the above requirements for the desired authoring tool suite, we developed the following design guidelines:

- Recognize the division between COTS tools and custom tools. Develop the custom tools as an integrated application. Develop consistent mechanisms for linking COTS tools' products with custom tools' data.
- Organize the custom tools in the unified application as content-specific screens to be enabled/disabled and/or restricted to fit the different requirements and allowed capabilities of different user categories.
- Provide consistent conventions across the different custom authoring tool screens. Especially address the pervasive need to manage large collections of cross-linked items, and to move fluidly among different data elements with a standard hyperlinking and history mechanism.
- Provide a mechanism to validate entered data, report potential issues, and facilitate access to tool screens and elements to correct those issues—extending with hyperlink and history mechanisms.
- Lay the groundwork for multi-user authoring, but ensure robust support for a single user in the first round.

Driven by these commitments, we developed a Java desktop application, based on the open source Java Eclipse Rich Client Platform (RCP). RCP offered several advantages for our purposes:

- It is a rich, robust, and portable framework. Eclipse RCP provides the infrastructure for the Eclipse Integrated Development Environment (IDE)—an immensely complex, extensible, cross-environment, industrial quality, yet open source, tool suite. It offers a strong platform on which to build, while also fitting the project constraint of delivering code without proprietary encumbrances.
- RCP offers deep integration with the Eclipse Modeling Framework (EMF), which support graphical design and automated code generation. This is an efficient way to work, and also fit the project constraint of delivering extensive design documentation to go along with the unencumbered code base.
- EMF also auto-generates substantial code for data persistence (on the back end) and for user interface integration (on the front end). The RCP graphical user interface (GUI) widget toolkit includes data binding

capabilities for EMF data structures. The persistence options allowed us to quickly build the single-user file-based version, while maintaining the option to later switch to a multi-user database-backed version.

Figure 1 shows a screenshot of the resulting tool. Beneath the window's title bar is the toolbar. The first toolbar widget shows that the current user has "scripter" level privileges—meaning access to essentially all the tool's capabilities. Other toolbar widgets support saving data, and moving back and forth in the object browsing history. Below the toolbar is the tab bar, offering access to the nine major screens of the integrated tool suite. To maximize ease-of-use, most of these screens share a family resemblance with the one shown in the figure. For most tabs, the bulk of the screen is given over to an editor, composed of a *collection view* on the left, and a *detail view* on the right.

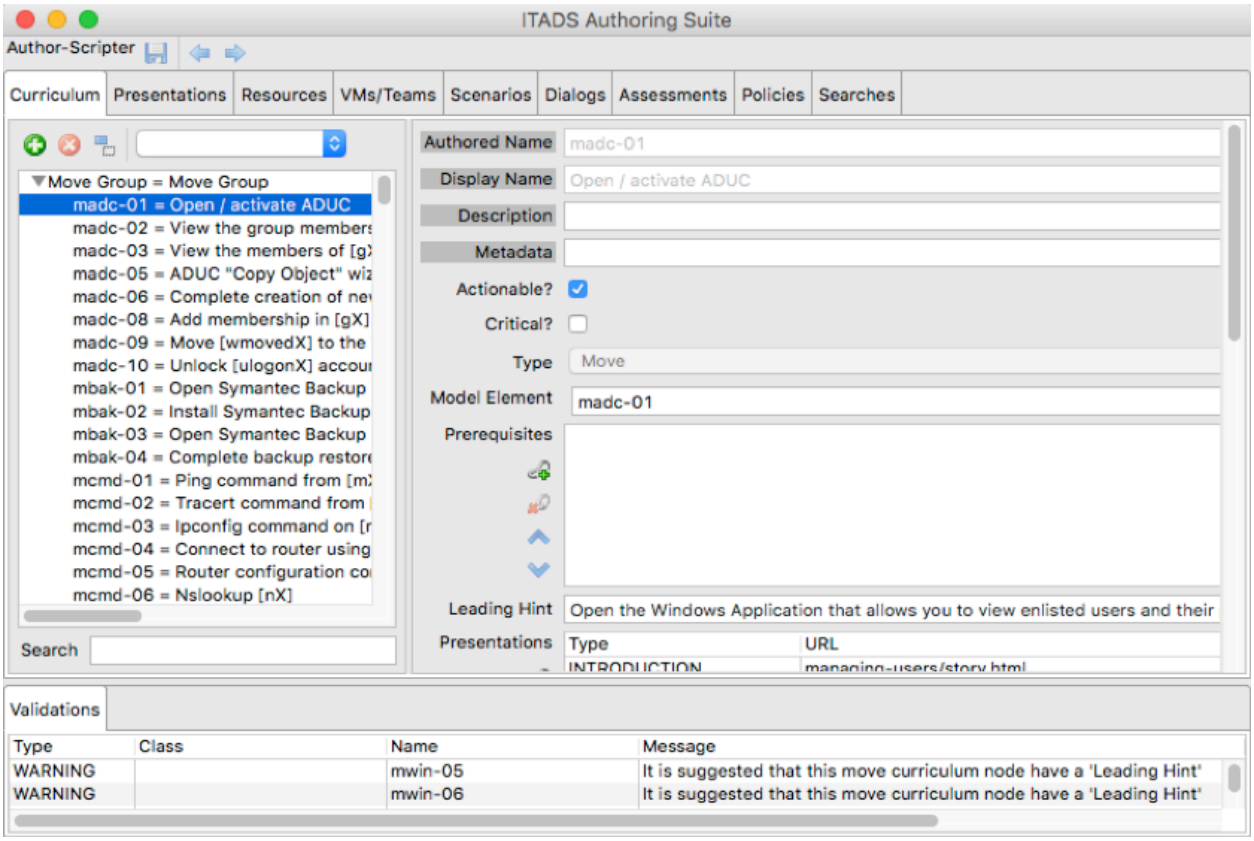

Figure 1. Sample Screen from ITADS Authoring Tool

The collection view shows a tree or list of entities managed in the tab. Figure 1 shows the Curriculum tab with its collection of curriculum nodes, in a tree reflecting their hierarchical organization. Most collection views include a drop-down list for filtering view contents by item type. They also include a type-in search box to enable filtering the view's contents by substring match. The collections typically also provide buttons to create, delete, and copy elements in the collection. Note that if the standard collection pane filtering and search mechanisms are insufficient, a user can go to the Searches tab, which provides a more elaborate set of mechanisms to find objects of interest.

The detail view is generally a property sheet with a consistent set of conventions for managing different data types, and especially for links to other objects (often defined in other tabs of the integrated authoring application). For instance, the "Prerequisites" field can contain a list of links to other curriculum nodes, while the "Presentations" field can contain a list of links to instructional presentations, defined in the Presentations tab. Clicking the link button associated with such a field yields a chooser dialog that lists appropriate objects, and provides the same type of substring-based filtering as the search field of collection views. Double-clicking on an element in one of these lists will shift the focus of the tool to the appropriate tab and item (while making an entry in the browsing history to allow easy return to the starting point).

Finally, at the bottom of the screen—consistent across all tabs—is the Validations area. Here the system displays warning and error messages based on its review of authored data. Again, double-clicking on a validation entry hyperlinks to the relevant tab and item where the data issue was identified and can be fixed.

The basic scheme described so far must be elaborated in the case of the more complex content types. For instance, Scenarios have so much associated data that a single detail page is insufficient. So the detail display for Scenarios is broken into six separate thematically organized sub-pages. Likewise, Dialogs, as tree-structured scripts with room for substantial variation at each node, require a basic property sheet be supplemented with a pair of nested pages.

The most complex tab is the one for Assessments, where the system's expert domain model is authored. Effectively, the Assessments tab manages an entire authoring sub-suite, organized into a set of six sub-tabs, each following the pattern of the major tabs already discussed. The six Assessments sub-tabs reflect a basic troubleshooting ontology, providing sub-tabs for Problems, Moves, Questions, Tasks, Parameter, and Findings. Among these, the greatest complexity is associated with the moves (and their sub-type, questions), and to a lesser extent with tasks. An overview of this system's assessment model and the logic encoded with moves is presented in (Jensen et al., 2016).

AUTHORING TOOL EVALUATION

Evaluation Goals

The evaluation of authoring tool suite aimed to answer the following questions:

- Can the tools be mastered and successfully used by Navy schoolhouse instructors?
- How happy are the intended users with the tools and what do they think about their prospects for adoption by the larger instructor community?
- What are the most important areas for improvement?
- How do these new authoring tools stack up against traditional training development tools?

Evaluation Method

The first three of the above questions were addressed in a pilot evaluation study in which we observed and surveyed representative instructors as they learned and used the authoring tools. The intent was to collect user feedback that could (in)validate existing designs and guide future design revisions, rather than to prove statistically significant findings (even testing with the entire cadre of instructors at the school would likely not suffice for that purpose). The final question was addressed by attempting to compare the cost structure and usage constraints of the new authoring capabilities with those of prior approaches.

Evaluation Study Participants

The subjects in this study were four instructors from the IT A-School—3 men and 1 woman, with between 2 and 6 years of experience teaching in the school. Three of the four had extensive Fleet IT experience, ranging from 6 to 18 years (the fourth had 8 years IT experience with the Air Force). All had prior military experience, ranging from 8 to 25 years, though only one was currently on active duty. Three of the four had been involved in project workshops at the design and implementation phase (though there was relatively little discussion of the authoring tools relative to the runtime, and the instructors did not have any chances to use of the authoring tools during those workshops).

Subject varied in their experience developing course designs, instructional materials, labs, exercises, computer-based training and simulation based training. One subject claimed almost no experience with any of these tasks. The rest claimed between half a year and 3 years experience for most tasks. However, only one subject claimed any experience developing simulation-based training. All subjects felt comfortable with PowerPoint, none with the tool used to convert PowerPoint slides for the web. There was also little familiarity with the VM management tools.

Evaluation Study Materials

During the training segments, subjects were provided with briefings, demonstrations, and detailed instructions on each major aspect of the authoring suite. They were given access to the relevant software, and presented with a set of structured authoring tasks. During those tasks, coaching and answers to questions were provided as needed. During

the evaluation segments, subjects were given a set of brief task statements asking them to carry out authoring tasks similar to those covered in the earlier training. They had access to the detailed task instructions from the training period. Where it would speed up their work and keep them focused on the tools (versus on inventing novel training content), pre-written content was provided (as had been the case in the training segments). Prompting was kept to a minimum during evaluation tasks. Questionnaires were interleaved with the various training and evaluation segments.

Evaluation Study Procedure

Schedules and timing of activities varied due to subject performance, experimenter availability, and evaluation process learning curve. To start, subjects were given ITS orientation sessions in pairs (roughly 3 hours long). Each subject then got individual training on the authoring suite (over a period of 2 days). The training mixed briefings, demonstrations, and guided task practice, along with time for questionnaires and unstructured interviews. Two experimenters were used—one as teacher and guide, the other as observer and recorder. Other project staff were on call to answer detailed questions, especially for issues related to the COTS tools.

The evaluation segment was a separate final day for each subject. Subjects were given a set of six brief task statements. Those six tasks mirrored things they had been trained to do over the preceding training days. During these tasks, observers restricted the comments and advice offered to what was strictly required to correct missteps. Following the evaluation tasks, the subjects were given the same questionnaires as from the earlier days, to see if their opinions had shifted with the additional exposure to and use of the system. Final comments were also solicited in an unstructured interview. Ratings reported below are based on the second/final pass through the questionnaires.

Evaluation Study Results

Performance Results

All subjects completed the training sessions within the planned time, most substantially faster than anticipated. The task guide format gave most subjects some difficulty, since it ended up as an amalgam of detailed example instructions and general reusable procedures. Nonetheless, all were able to complete the entire set of guided tasks. All subjects also completed the six evaluation tasks well within the allotted day. The following table presents approximate minimum, maximum, and average timings (as hours:minutes) for each evaluation task.

Table 1. Evaluation Tasks and Timings

Task	Min	Max	Avg
Add slides to an existing presentation	0:15	1:00	0:30
Create a new presentation	0:20	0:35	0:30
Modify an existing resource	0:05	0:20	0:10
Create a variant of a procedural scenario	0:45	1:25	1:10
Create a variant of a diagnostic scenario	0:40	1:15	0:55
Create a new dialog-based scenario	0:30	0:35	0:30
Total	**2:35**	**5:10**	**3:45**

An analysis of errors revealed two recurring sources of difficulty experienced by authoring subjects: (1) Forgetting to carry out steps in procedures, particularly steps required by ITS-specific conventions for using COTS tools; (2) Misunderstanding the given dialog formats, especially how to use some of the more advanced features of dialogs. We believe that these issues can be addressed through a combination of better documentation and training, and the introduction of further simplifications, such as scripts to handle some aspects of COTS tools data preparation.

Subjective Feedback

We also solicited and received subjective feedback from our subjects. Our formal questionnaires asked subjects to rate their agreement with a set of statements on a scale ranging from 1 = “Strongly Disagree” to 5 = “Strongly Agree.” The following table presents average scores, giving a sense of the overall positive reaction to the tools.

Table 2. Evaluation Ratings

Statement	Avg.
The Authoring Tool (AT) is logically organized.	4.75
The AT behaves as I expect it to (is predictable).	4.50
The AT performs reliably (doesn't have bugs).	4.25
The AT is responsive (runs fast enough).	4.75
COTS authoring of presentations, resources, and inbox items is natural and simple.	4.50
COTS authoring of VM teams is natural and simple.	4.50

Comparisons with Prior Tools

The newly developed tools' strengths are (1) they allow creation of new kinds of dynamic and adaptive content and (2) they can embed guidance to help authors build content that aligns with the ITS's specific (and effective) pedagogical approach. The tools' weaknesses compared to existing authoring tools are (1) they are prototypes (and thus lack polished user documentation, integrated help, formal training, and some yet-to-be-developed features), and (2) they are relatively complex (or at least novel). Internal assessment suggests that content development costs are roughly on par with those reported in the most authoritative study of large-scale production of military interactive multimedia instruction (Fletcher, 2010). ITADS accomplishes this while providing more capability and flexibility. More capability means that this ITS and its authoring tools support adaptive simulation-based instruction. More flexibility means that most aspects of the system's more complex behavior can be modified using authoring tools.

With respect to ITS authoring more specifically, the standard estimates for existing models or rule -based systems are in the range of 200–300 hours of development per hour of instruction (Aleven, et al., 2009). However, the underlying sources give little detail—e.g., categories/costs of laborers used or exactly what was produced. The most detailed source noted that developing an underlying production system averaged 10 hours or more for each production rule (Anderson, et al., 1995). Our system is not based on production rules, so direct comparison to this benchmark is difficult. In any case, if the old ITS effort estimates are based primarily on rule development time, there is much more to preparing an hour of experiential adaptive instruction than building the assessments (rules or otherwise), and Artificial Intelligence PhDs are not necessary for many of those tasks. More recent work on a limited form of intelligent tutoring—eschewing any significant reasoning—claims up to a 4x speed increase over the notional 200–300 hours/hour benchmark; this is combined with an estimated 2x labor-cost decrease, to yield a potential 8x improvement over "traditional intelligent tutoring system" authoring (Aleven, et al., 2009). Again, direct comparisons are difficult.

From the Government's perspective, the key cost issue is avoiding vendor lock-in. Any content task that can be done in-house at a Navy school is a piece of work that: (a) does not require a contract to be let, managed, audited, etc. with all the attendant costs in both time and money; (b) will be performed at a cost commensurate with its difficulty, rather than based on what the vendor believes they can charge given a lack of credible alternatives; and (c) may be performed more efficiently by an instructor who better and more quickly understands the situation in the schoolhouse that motivates the change. The Navy owns all code for the ITADS runtime and custom authoring suite.

While the described system is by no means a finished product, already we have demonstrated that Navy instructors can easily make changes and extensions to the didactic instructional materials. Likewise, they can comfortably edit and extend many of the tutor utterances such as hints and dialogs. With a bit more effort they can create variants of existing scenarios (with their supporting simulation configurations). With deeper knowledge of the assessment model than we could impart during our authoring trials, they should also be able to create entirely new scenarios that exploit the existing assessment structure. Thus, there are a wide range of useful changes already supported by the authoring suite, many of which can be carried out without requiring a new contract with the original vendors.

RELATED RESEARCH

There has been substantial work on ITS authoring tools (Murray, 1999). Our authoring tool work addresses two themes we have called tiering and bootstrapping (Domeshek, Jensen, & Ramachandran, 2015). *Tiering* is the division of content by type—e.g., didactic content, versus scenario configuration, versus assessment logic, etc.—allowing

authors with different knowledge and skills to create those kinds of content for which they are most qualified (using different, tailored, tools). *Bootstrapping* is a development process where the cost of authoring is reduced over successive spirals, by starting with scenario-specific content and incrementally generalizing to create more reusable components.

Several approaches related to tiering have appeared in the literature. For instance, the meta-authoring approach allows a class of skilled authors to use generalized authoring tools to generate more specialized authoring tools for use by lower-skilled authors working in specific domains or pedagogical styles (Qiu & Riesbeck, 2005; Hsieh, Halff, & Redfield, 1999). There has also been work on specialized tools for particular content types. For instance, (Nye, et al., 2014) describe a tool that uses a tiered approach for augmenting web content with AutoTutor like dialogs.

Bootstrapping has been gaining traction in recent years, primarily focused on application of machine learning (Kumar, Roy, Roberts, & Makhoul, 2014; Aleven, McLaren, Sewall, & Koedinger, 2006). One approach starts with handcrafted knowledge, but then uses student performance data to improve a tutor's assessment or student modeling knowledge (Baker, Corbett, & Aleven, 2008; Barnes & Stamper, 2008). Such bootstrapping stays within the confines of the modeling approach—i.e., ITSs that start out example-based remain example-based. In contrast, our by-hand analysis evolved both the accuracy of the model and the form of the modeling representation itself.

CONCLUSION AND LESSONS LEARNED

Our experience building and testing the described authoring tools suggests the following conclusions and lessons:

- Using this authoring suite, military instructors with little programming experience and given limited training were able to successfully create and modify content. Thus, it is clearly possible to build usable broad-spectrum authoring tools relatively efficiently. It helps to emphasize consistent, straightforward forms-based design. It also helps to leverage power-tools, such as RCP and EMF. We did not find great need for fancier graphical views (e.g., arbitrary network graphics). We did make productive use of custom form design—e.g., object and field clustering, ordering, nesting, and widget choice. We also provided many useful content-sensitive interaction conventions—e.g. collection searching and filtering, object linking both automatic and user-driven (from system-generated candidate lists), hyperlink content browsing and history management, data validation, and guards against large-scale destructive changes.
- The use of COTS applications for some content preparation is nearly inevitable for media, and quite likely for pre-existing simulation components. It can lower implementation costs, and may improve usability (though less so if the COTS tools are unfamiliar to the intended user population). However, integration of COTS tools with custom tools may require more effort than would initially be apparent. COTS tools may have to be used according to special conventions to produce consistent products that work with the ITS. Such conventions need to be invented, documented, and taught. Scripts may need to be developed to drive the tools or process their input/output files to handle repetitive aspects of the ITS use conventions.
- We saw that custom authoring tools, built to emphasize high usability and low implementation costs through consistency and simplicity of design, can do a good job of supporting rapid, structured, validated data entry. However, our experience does not yet support their adequacy for the creative aspects of content development. In particular, the *design* of scenarios and models—inventorying and choosing problems, generating plausible hypotheses, and identifying student moves to differentiate among hypotheses—was primarily supported by a set of on-line shared spreadsheets, because they were flexible and available.
- Our inability to develop and prove custom authoring tools for scenario design reflects, in part, the need for stable content formats prior to building custom tools. Unfortunately, scenario conceptualization tends to happen early, and the formats used are often subject to change, driven by the needs of new scenarios. Spreadsheets filled the gap as an especially flexible kind of COTS tool. It is worth considering explicit adoption of spreadsheet use, not just as a stopgap, but as a formal part of scenario data development (Domeshek, 2009). As with use of other COTS tools, this can usefully be complemented by development of conventions, templates, scripts, and data import tools. Scenario design work can then potentially go on in a reliable COTS spreadsheet while these ancillary tools evolve to track changing usage patterns.
- Another way to improve custom tools' utility for more creative/exploratory authoring is to integrate content testing capability directly into the authoring tools. For example, we were able to integrate the dialog runtime into the dialog authoring tab. This made it much more fluid to experiment with different ideas about dialog

structure and content, and immediately see results. We were unable to do this for the larger simulation environment because the full VM environment was too heavy-weight, and tools for managing the VMs were late to mature.

- The difference between single-user tools and multi-user tools is substantial, though the right power tools (e.g., for Java, EMF and Hibernate) can ease the transition. The difference between desktop applications and web applications is probably even greater. There may be more tools that try to bridge that gap, but none are mainstream and none make it truly easy. The web client/server and browser models intrude, and the gap between Java (or Java-embedded languages) and JavaScript requires cross-compilation which complicates development.

We envision two main thrusts of research and development building on the above-described work and lessons:

- First, we would like to continue developing and evaluating the described system—both authoring and runtime components. The most immediate high-value authoring improvements for the current application would be tooling to manage the VM-based simulation environment. However, more critical long-term issues fall under the general theme of "scale:" (a) increasing the number of scenarios authored; (b) transitioning to multi-user concurrent and distributed authoring; (c) evaluating authoring with a larger group of instructors; and (d) exploring application of the core system logic to a range of different training domains. We expect that considering troubleshooting not just computers and networks, but other kinds of technical systems as well, would drive us to generalize, further test, and strengthen many aspects of ITADS machinery, improving both authoring and runtime components.
- Second, we would like to explore techniques for rapid and flexible authoring tool development aimed at supporting the more creative, fluid, and early-stage scenario design tasks we were unable to address in this work. Such tools would ideally be as robust and flexible as the COTS spreadsheet applications we actually used, but with a much greater ability to capture the evolving syntax and semantics of new application domains. Tools able to rapidly adapt to novel syntax and semantics could help authors in several ways: (a) clarifying and documenting evolving ideas about scenario structure; (b) speeding scenario generation by reducing by-hand data-entry labor; (c) validating entered data to more quickly catch inconsistencies, redundancies, and errors; and (d) helping track and manage the additional authoring work required to turn initial scenario sketches into full executable scenario specifications.

ACKNOWLEDGEMENTS

This work was performed under a contract awarded and administered by the U.S. Navy, Naval Air Warfare Center Training Systems Division (NAWCTSD) in Orlando, FL for the Navy's Center for Information Warfare Training (CIWT) IT A-School in Pensacola, FL. The authors wish to thank the entire ITADS team at Stottler Henke Associates Inc and Comtech Telecommunications Corp., as well as our Government sponsors, for their dedication to making this project a success.

REFERENCES

Aleven, V., McLaren, B., Sewall, J., & Koedinger, K. (2006). The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. *Intelligent Tutoring Systems Lecture Notes in Computer Science, Vol. 4053*, 61-70.

Aleven, V., McLaren, B.M., Sewall, J., & Koedinger, K.R. (2009). A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, **19**(2), 105-154.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive Tutors: Lessons learned. *The Journal of the Learning Sciences, 4,* 167-207.

Baker, R., Corbett, A., & Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. *Intelligent Tutoring Systems Lecture Notes in Computer Science, Vol. 5091*, 406-415.

Barnes, T., & Stamper, J. (2008). Toward automatic hint generation for logic proof tutoring using historical student data. *Intelligent Tutoring Systems Lecture Notes in Computer Science, Vol. 5091*, 373-382.

Bloom, B. (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).

Domeshek, E., Holman, E., & Luperfoy, S. (2004) Discussion Control in an Automated Socratic Tutor. *Proceedings of the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC 2004).*

Domeshek, E. (2009) Scenario-Based Conversational Intelligent Tutoring Systems for Decision-Making Skills. *Proceedings of the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC 2009).*

Domeshek, E., Jensen, R., & Ramachandran, S (2015) Tiering, Layering and Bootstrapping for ITS Development. In R. Sottilare, A. Graesser, X. Hu, and K. Brawner (Eds.), *Design Recommendations for Intelligent Tutoring Systems: Volume 3 – Authoring Tools and Expert Modeling Techniques* (pp. 335-346). Orlando, FL: U.S. Army Research Laboratory.

Fletcher, J.D. (2010). Research Foundations for the Advanced Distributed Learning Initiative. Institute for Defense Analysis (IDA). IDA Document D-4118.

Hsieh, P.Y., Halff, H.M., & Redfield, C.L. (1999). Four easy pieces: Development systems for knowledge-based generative instruction. *International J. of Artificial Intelligence in Education (IJAIED)*, 10, 1-45.

Jensen, R., Ramachandran, S. Domeshek, E. Tang, K., & Marsh, J. (2016). Simulation Awareness: Assessing Performance with Limited Simulation Instrumentation. *Proceedings of the Interservice / Industry Training, Simulation, and Education Conference (I/ITSEC 2016)*.

Kulik, J. A., & Fletcher, J. D. (2016). Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of Educational Research*, *86*(1), 42-78.

Kumar, R., Roy, M. E., Roberts, R. B., & Makhoul, J. I. (2014). Towards Automatically Building Tutor Models Using Multiple Behavior Demonstrations. *Intelligent Tutoring Systems Lecture Notes in Computer Science, Vol. 8474*, 535-544.

Ma, W., Adesope, O. O., Nesbit, J. C., & Liu, Q. (2014). Intelligent tutoring systems and learning outcomes: A meta-analysis.

Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International J. of Artificial Intelligence in Education (IJAIED),* **10**, 98-129.

Nye, B. D., Rahman, M.F., Yng, M., Hays. P., Cai, Z., Graesser. A., & Hu, X. (2014). A tutoring page markup suite for integrating shareable knowledge objects (SKO) with HTML. *Proceedings from Intelligent Tutoring Systems (ITS) 2014 Workshop on Authoring Tools.*

Qiu, L., and Riesbeck, C. (2005). The design for authoring and deploying web-based interactive learning environments. *World Conf. on Educational Multimedia, Hypermedia and Telecommunications*.

Salas, E., Milham, L. M., & Bowers, C. A. (2003). Training evaluation in the military: Misconceptions, opportunities, and challenges. *Military Psychology*, *15*(1), 3.

Smith, R. (2014). Military Simulations Using Virtual Worlds. *Mark Grimshaw (Hg.): The Oxford Handbook of Virtuality. Oxford*, 666-679.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, *46*(4), 197-221.