Assessment and Content Authoring in Semantic Virtual Environments

Christian Greuel, Karen Myers, Grit Denker, Melinda Gervasio
SRI International
Menlo Park, CA
{firstname.lastname}@sri.com

ABSTRACT

Virtual environments (VEs) provide an appealing vehicle for training complex skills, particularly for domains where real-world practice incurs significant time, expense, or risk. Two impediments currently block widespread use of intelligent training tools for VEs. The first impediment is that techniques for assessing user performance focus on algorithmic skills that force learners to follow rigid solution paths. The second impediment is the high cost of authoring the models that drive intelligent training capabilities.

This paper presents an approach to training in VEs that directly addresses these challenges and summarizes its application to a weapons maintenance task. With our approach, a learner's actions are recorded as he completes training exercises in a semantically instrumented VE. An example-tracing methodology, in which the learner's actions are compared to a predefined solution model, is used to generate assessment information with contextually relevant feedback. Novel graph-matching technology, grounded in edit-distance optimization, aligns student actions with solution models while tolerating significant deviation. With this robustness to learner mistakes, assessment can support exploratory learning processes rather than forcing learners down fixed solution paths.

Our approach to content creation leverages predefined ontologies, enabling authoring by domain experts rather than technology experts. A semantic mark-up framework supports authors in overlaying ontologies onto VE elements and in specifying actions with their effects. Drawing on these semantics, exercises and their solutions are created through end-user programming techniques: a domain expert demonstrates one or more solutions to a task and then annotates those solutions to define a generalized solution model. A concept validation study shows that users are comfortable with this approach and can apply it to create quality solution models.

ABOUT THE AUTHORS

Christian Greuel is a Senior Research Artist in the Computer Science Laboratory at SRI International, with a focus on visual computing. His areas of applied research include design of tools, techniques, and processes for representation of and interaction with virtual environments; web-based training applications; geographic visualization; geometric modeling; multi-modal data representation; and immersive audio. He earned an MFA in Performing Arts Design and Technology from the California Institute of the Arts.

Karen Myers is a Principal Scientist in the Artificial Intelligence Center at SRI International, where she conducts basic and applied research in intelligent systems. Her specific areas of expertise include autonomy, intelligent assistants, mixed-initiative planning and control, and programming by demonstration. In addition to being widely published, she has overseen several successful transitions of her research into use both by the U.S. Government and within commercial settings. She has a Ph.D. in Computer Science from Stanford University.

Grit Denker is a Program Director in the Computer Science Laboratory at SRI International. Her expertise is in automated reasoning applications using semantic technologies, ontologies, and rules; cognitive and policy-based systems; specification and verification of security, network and privacy policies in distributed systems; and novel human-machine interface technology. She has a Ph.D. in Computer Science from the Technical University of Braunschweig, Germany.

Melinda Gervasio is a Principal Scientist in the Artificial Intelligence Center at SRI International, where she pursues research in adaptive intelligent systems. Her areas of expertise include intelligent assistants, end-user programming, adaptive personalization, and interactive learning. More recently, she has focused on developing intelligent systems that learn from and interact with humans for mixed-initiative problem-solving. She has a Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign.

Assessment and Content Authoring in Semantic Virtual Environments

Christian Greuel, Karen Myers, Grit Denker, Melinda Gervasio
SRI International
Menlo Park, CA
{firstname.lastname}@sri.com

INTRODUCTION

Proficiency in complex skills improves with practice. Virtual environments (VEs) provide an appealing vehicle for training complex skills, particularly for domains where real-world practice incurs significant time, expense, or risk. Currently, two impediments block widespread use of intelligent training tools for virtual environments. The first is that most current techniques for assessing user performance force learners to follow rigid solution paths. These techniques limit the applicability of the technology to domains with 'algorithmic' solutions, whereas many real-world training domains have a more open-ended nature. The second is the high cost of authoring the models that drive intelligent training capabilities. In particular, authoring must be done by highly trained experts who have deep familiarity with the representation and reasoning mechanisms employed by the assessment capabilities.

This paper presents an approach to training in VEs that directly addresses these challenges. With our approach, a learner's actions are recorded as he completes training exercises in a semantically instrumented VE. An example-tracing methodology, in which the learner's actions are compared to a predefined solution model, is used to generate assessment information with contextually relevant feedback. Novel graph-matching technology, grounded in edit-distance optimization, aligns student actions with solution models while tolerating significant deviation. With this robustness to learner mistakes, assessment can support exploratory learning processes rather than forcing learners down fixed solution paths.

Our approach to content creation leverages predefined ontologies, represented in a standards-compatible language. A semantic mark-up capability supports authors in specifying semantic overlays based on these ontologies to define types and attributes of VE components along with relationships among them. These overlays, coupled with semantic instrumentation of user actions, enable *understanding* of student activity in a VE in terms of semantic concepts that are well aligned with pedagogical constructs.

Drawing on these semantics, exercises and their solutions are created through end-user programming techniques. In particular, a domain expert *demonstrates* one or more solutions to a task, and then *annotates* those solutions to define a generalized solution model. These tools are specifically designed to enable content authoring by subject matter experts (SMEs) rather than technology experts. A concept validation study shows that users are comfortable with this approach and can apply it to create quality solution models.

We have developed a prototype system called Semantically-enabled Assessment in Virtual Environments (SAVE) that combines these capabilities into a flexible training framework that can provide assessment feedback for complex training tasks. SAVE has been applied to training tasks related to weapons maintenance (specifically that of an M4 series rifle carbine), drawing on requirements derived from a U.S. Army soldier training publication (U.S. Department of the Army, 2012). The assessment piece of SAVE has also been applied to support training the use of Command Post of the Future (CPOF), a collaborative geospatial visualization environment and planning tool used extensively by the U.S. Army (Myers et al., 2013).

This paper begins with an overview of our approach. We then proceed to describe elements related to the *learner experience*, namely the process by which a training exercise is completed and assessment information is generated. Next, we describe our approach to *content development*, covering the specification of semantic layers for a VE, the creation of exercises, and the authoring of exercise solutions. We summarize results from a user study that show that subjects who are unfamiliar with the underlying technology are both comfortable with our solution authoring approach and capable of applying it to generate quality solution models. We next describe our application of SAVE to the M4 specific training problem, discuss related work, and then close with a summary of our contributions.

OVERVIEW OF APPROACH

SAVE supports two classes of end users: the *learners* who perform exercises in the VE and receive assessment feedback, and the *content authors* responsible for developing exercise materials. Figure 1 presents the overall architecture of the SAVE training framework, organized around these two user types.

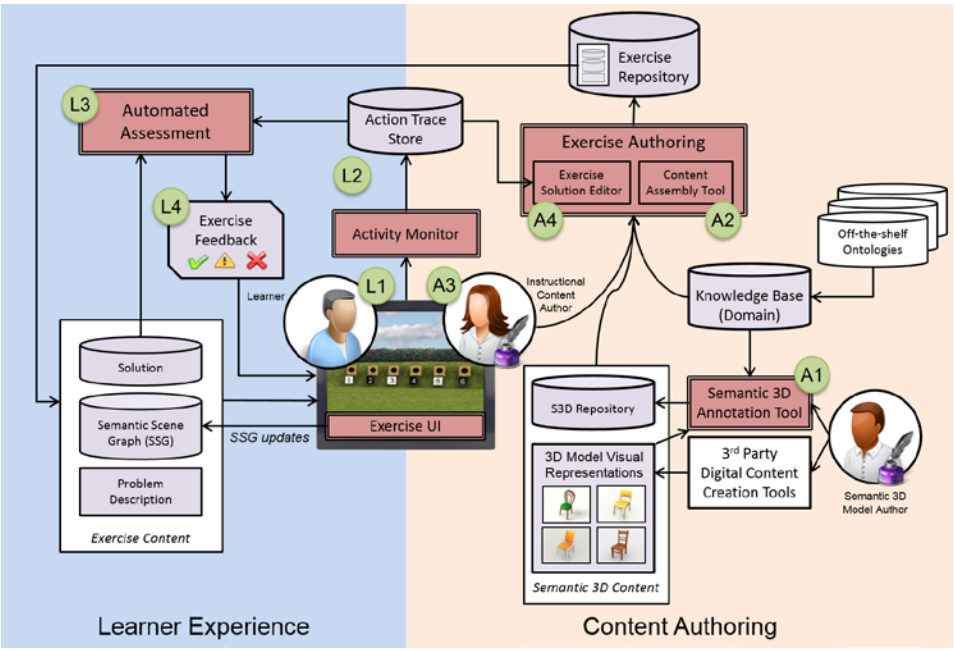

Figure 1. Overview of the SAVE framework, which supports learners and content authors.

The exercise user interface (EUI) provides both a visualization of a training problem and the interactive means by which the learner can perform actions to solve the problem (L1 in Figure 1). The training problem is presented in terms of a task to be completed by the learner and a specification of the initial scene in which the exercise is to take place. This scene description is captured in a structure referred to as a *semantic scene graph* (SSG). The actions performed by the learner in the EUI result in updates to the SSG. The actions are also logged in a semantically grounded trace (L2). The automated assessment module compares these actions to the solution model for the problem (L3), and generates learner feedback (L4) that is displayed in the EUI.

The content that drives exercises is created through a collection of authoring tools, utilized by two different types of content developers, working in collaboration. Starting with previously defined 3D models, a *semantic 3D (S3D) model author* uses an annotation tool to define overlays (A1 in Figure 1) that link nodes of each model with classes in predefined ontologies, thus providing a semantic characterization of the objects and their components. These overlays are added to a repository of available S3D assets.

With a content assembly tool, an *instructional content author* uses these semantically labeled objects to compose a specific VE for a training exercise (A2). The author uses a set of complementary authoring tools to define the range of allowed solutions to a training exercise. A key element of our approach is that the author first defines the procedural structure of the solution by demonstrating how it should be done in the EUI (A3); a companion solution editor (A4) then supports the author in specifying annotations that define allowed variations from the demonstration.

LEARNER EXPERIENCE

Exercise Completion

The EUI loads exercise-specific content, displays 3D models (1 in Figure 2) of the objects in the initial configuration for the exercise, and provides the learner with affordances such as context menus (2) to interact with these objects.

Ontologies and rules are used to form a semantic characterization of an exercise domain, the actions that are possible within that domain, and the preconditions and effects of those actions. Ontological classes are mapped to appropriate geometric nodes of the 3D model, thus semantically instrumenting the VE.

Figure 2. The EUI allows the learner to perform an exercise, such as M4 maintenance, in a virtual environment and receive assessment feedback.

A query mechanism supports communication between the frontend graphical representation of the scene and its corresponding backend semantic representation. The EUI is aware of what virtual objects can be instanced and how each should be displayed, while the ontology characterizes these objects and the actions that can be performed on them. The concerns of rendering and semantic definition remain separate; the EUI does not need to track knowledge about components, and conversely, the ontology does not need to be aware of the 3D scene display.

The activity monitor collects the learner's direct interactions with the EUI (e.g., in terms of mouse clicks and keyboard input) and converts them into an action trace—a semantic characterization of the learner's activities that is grounded in the ontology. These action traces serve two purposes within SAVE, depending on whether the user is a learner or an instructional content author. For a learner, the action trace is the basis for the automated assessment of their performance for that exercise. For an author developing a new exercise, the action trace provides the procedural basis that serves as a starting point for the exercise solution model.

Automated Assessment

The automated assessment module in SAVE builds substantially on SRI's Drill Evaluation for Training (DEFT) technology (Myers et al., 2013). DEFT was developed as part of a pilot project on automated assessment of skills for soldiers learning to use CPOF—a collaborative geospatial visualization and planning environment used extensively by the U.S. Army for planning military operations. Below, we describe the technical foundations for the automated assessment capability, followed by a discussion of our approach to presenting assessment information to a learner.

Assessment is based on a comparison of the semantic action trace generated by the instrumented VE to a pre-specified *exercise solution model* that defines the range of acceptable solutions for the exercise. This comparison is used to create assessment information with contextually relevant feedback for the learner: identifying conceptual errors or mistakes, providing guidance in the form of hints to help complete a task, and suggesting links to relevant training materials. The feedback is presented to the user in the EUI (3 in Figure 2).

Table 1: Annotations for Solution Specification

	Annotation	Semantics
Steps	Group	Group a set of related steps
	In any order	Perform a pair or group of steps in any order
	Optional	Step is not required as part of the solution
Parameters	Any of Set	Use any of an enumerated set of values (e.g., Strawberries, Blueberries, Raspberries)
	Any of Type	Use any of a given semantic type (e.g., Fruit, Vegetable)
	Range	Use a range of values (e.g., 1-5)
	Same	Use the same object in different steps

The exercise solution models are defined in terms of one or more *generalized action traces*, each consisting of (a) a sequence of *steps*, and (b) *annotations* that specify allowed variations. A step can be a parameterized action, a class of actions, or a set of options, each of which is itself a partially ordered set of steps. Annotations can be defined over individual steps (e.g., optionality of a step), sets of steps (e.g., step ordering constraints), and step parameters (e.g., a parameter must have a specific value or satisfy a property). Annotations support action ordering and grouping constraints; parameter type, value, and equality constraints; and state constraints, which capture requirements on the application state or on object properties that cannot be determined from actions themselves (see Table 1). These abstractions enable compact representations of large solution spaces, rather than having to explicitly enumerate all allowed ground solutions.

The automated assessment capability determines a mapping from the student's response to the exercise solution model. This alignment problem is formulated as *approximate graph matching*, using graph edit distance to rate the quality of the mappings. Graph edit distance measures the cumulative cost of graph editing operations needed to transform the student response into an instance consistent with the exercise solution model.

To use this graph matching approach, the exercise solution model is encoded as one or more graphs, each representing a family of possible solutions. Within the graphs actions and their parameters are nodes, parameter roles are links, and required conditions within the solution (e.g., action orderings, parameter values) are constraints. The student response is represented similarly as a response graph. Alignment involves finding the lowest-cost mapping between the response and a solution graph, with costs incurred for missing mappings and violated constraints. The intuition is that the lowest-cost alignment corresponds to the specific solution the student is most likely attempting. From this alignment, an assessment is generated that identifies differences between the response and the exercise solution model, which translate to specific errors the student has made (e.g., out-of-order, missing, or extra actions; incorrect parameter values) and to the corrections needed. Because the exercise solution models and learner responses are captured at a semantic level of abstraction, the feedback from the automated assessment process is readily understandable by the learner.

The visual display of assessment feedback in the original DEFT system presented the full action trace to the learner, with overlaid markings that identified learner mistakes. Feedback from users showed a preference for a summary view of their mistakes: they did not want to see everything they had done, rather only what they had done incorrectly (Myers et al., 2013). For this reason, we transitioned from the original trace-based presentation of feedback to a summary-based approach for SAVE (Figure 3). A further advantage of the more compact summary-based approach is that it can be presented as text within the EUI, thus eliminating the need for a separate assessment interface.

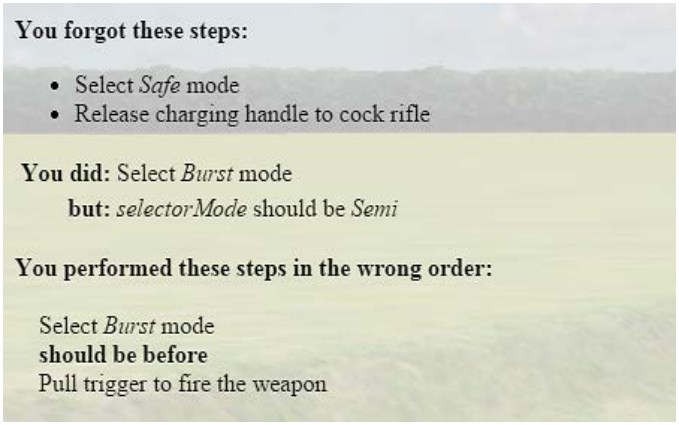

Figure 3. Sample summary-based assessment feedback for the M4 domain as displayed within the EUI.

Our approach to automated assessment is similar to that of example-tracing tutors (Aleven et al., 2009), which compare learner actions to a graph representing alternative solution paths. However, example-tracing tutors rely on an exhaustive enumeration of all possible solutions, which is feasible for domains with algorithmic solutions but not for domains with more open-ended tasks. In contrast to other example-tracing assessment capabilities, our assessment method does not restrict a user to a limited set of solution paths. Rather, the flexible matching provides tolerance to user mistakes, enabling assessment for domains in which more exploratory styles of task completion are an important part of the learning experience.

CONTENT AUTHORING

For intelligent training systems to be widely deployed, course developers must be able to easily construct the exercises in their curricula. To this end, we developed a set of tools for both the S3D model author, who semantically marks up the individual 3D assets, and the instructional content author, who assembles these assets into a complete exercise scene and specifies the exercise solution.

Exercise Content

3D models are geometric representations of objects that can be rendered in a virtual environment. The S3D model author uses an annotation tool (Figure 4) to create semantic overlays for these underlying 3D models. The tool is a graphical interface that enables authors to associate predefined ontological classes with corresponding geometric nodes of a 3D model without having to write code. These associations create links between the graphic representations of an object's components, such as the parts of a machine, and conceptual models describing semantic properties of these components.

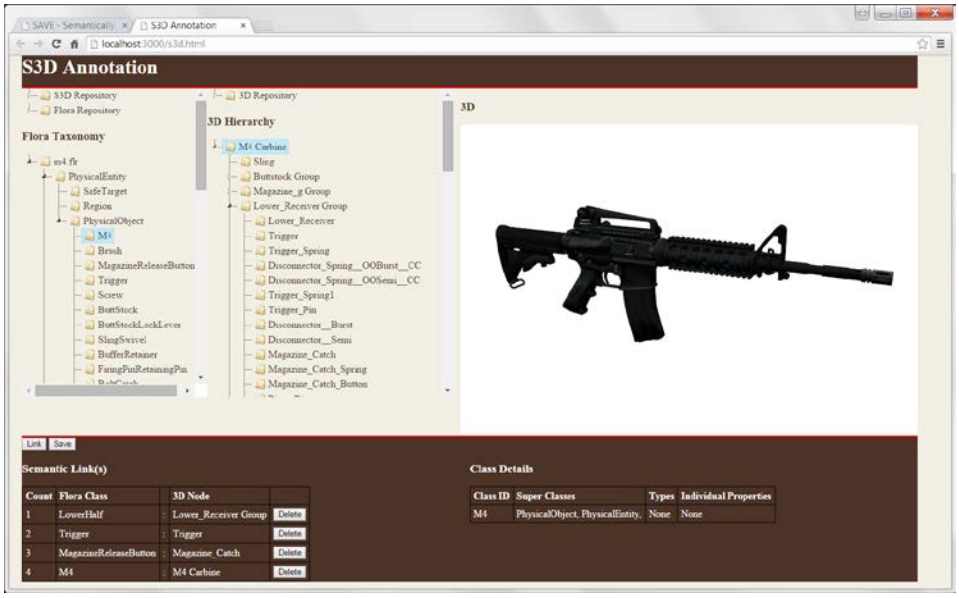

Figure 4. The S3D annotation tool is used to map semantic classes to specific nodes of a 3D model.

The overlays assign each element a semantic identity and associated properties that can be reasoned about by the automated assessment module. Properties may include information about relationships between various elements, such as hierarchical grouping. For example, an M4 is separable into two halves, known as an *upper* and a *lower*, which in turn are each made up of various subcomponents. The ontologies also hold information about what actions may be executed upon each component by the user as well as the effects of such actions.

As discussed earlier, a key feature of our approach to semantic instrumentation is that the knowledge base about the virtual objects exists separately from their 3D model representations. Thus the ontology can be extended or modified independently without the need to edit the 3D model file. As another benefit, the same ontology can also be mapped to different versions of the 3D model, e.g., one of higher or lower fidelity, or used in a different set of exercises.

Indeed, there is no reason that the semantic classes could not be mapped to alternate object representation modalities, whether 2D graphics, immersive audio environments, or even text-based worlds.

From the semantically annotated 3D models, the instructional content author constructs an exercise scene for an interactive training exercise using a content assembly tool (Figure 4). The author selects annotated models from the S3D repository and spatially arranges them in the VE to create an exercise scene. Additional objects that may be instantiated in the scene by the user during the exercise, such as tools stored on a shelf, are also declared. The scene represents the environment as it will be presented in the EUI when the exercise starts.

Exercise Solution

Within SAVE, solutions are defined as generalized demonstration traces that consist of (a) a sequence of steps to be performed, and (b) annotations that show allowed variability from those steps. Authoring exercise solution models by a content developer involves first demonstrating a specific solution to a training exercise and then specifying annotations that generalize from that single demonstration to the full range of allowed solutions. The first step is performed within the EUI; the second step is performed within the exercise solution editor (ESE). The output of this process is a collection of constraints, referred to as the exercise solution model, which is stored in an XML-based representation. This approach is in line with recommendations made by others that models should be created from data and interactions, to the extent possible rather than handcrafted (Brawner et al., 2012).

The ESE supports content developers in both viewing action traces from demonstrations performed in the EUI, and in adding annotations to traces to capture allowed generalizations. The design of the ESE was heavily influenced by a user study described in the next section. This study validated our general approach, showing that users can both understand traces and generalize them with appropriate annotations. The study also showed that users expect actions in a solution to be performed in the same order as in the demonstration trace, unless expressly indicated otherwise. For this reason, the ESE automatically generates ordering constraints in the exercise solution model based on the assumption that steps must follow the demonstration order unless expressly marked otherwise.

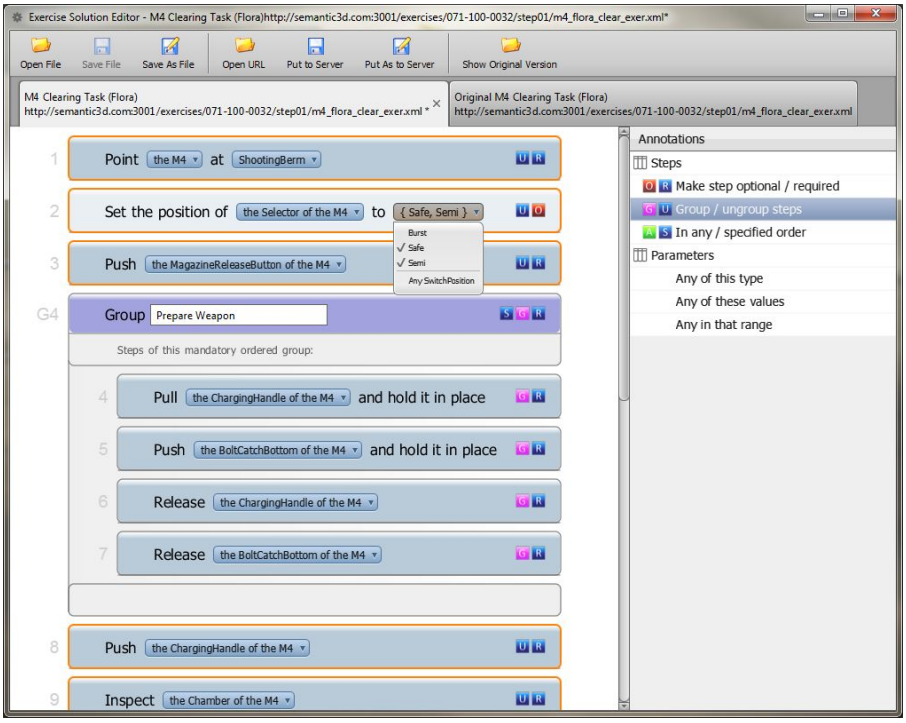

Figure 5. The ESE allows an author to generalize an exercise solution by editing and annotating it.

The user interface of the ESE (Figure 5) provides the visualization of the action trace as a sequence of steps in the left-hand pane, and a panel of available annotation operations in the right-hand pane. Each step in the action trace shows a textual description of the action as well as the action's parameters. Steps can be organized into groups and assigned meaningful names at the discretion of the content developer (e.g., the "Prepare weapon" group in the figure). The UI allows authors to perform most annotation-related operations by either selecting from actions in the panel or by clicking directly on an item (step or parameter) in the action trace to display a context-sensitive list of options. This multi-access support reflects findings from the user study, which showed differing preferences for the two interaction styles.

Icons on individual steps show the status for grouping and optionality. An author can toggle an annotation on/off by selecting the appropriate menu item or clicking directly on the associated icon on the step. Similarly, generalizations of parameters can be made through the appropriate menu item or by clicking on the parameter to access a context-sensitive set of generalization options. The use of color in icons and for parameter highlighting makes it easy for an author to quickly see the scope of annotations defined within a given solution.

USER STUDY ON SOLUTION AUTHORING

We conducted a concept validation study, summarized in (Myers and Gervasio, 2016), to provide an empirical basis for the design of our solution-authoring module. This study had two objectives: (a) to determine whether SMEs, rather than experts in intelligent training systems, can successfully apply our general approach to solution authoring, and (b) to elicit feedback on the design of a tool that would support their authoring needs. We conducted a two-on-one (facilitator + note-taker vs. subject) paper prototype study requiring approximately two hours for each subject. Because content authors will be experts in their application domains, we required a study domain for which subjects would have expert-level knowledge. We chose cooking for the study tasks, because people feel expert in how they like to prepare their food (even if their preparations may be personalized).

The subjects were asked to demonstrate the preparation of two simple dishes and to document how their demonstrations could be generalized while remaining valid for those dishes. After the subject completed the two tasks, the facilitator led an open-ended discussion with the subject about what was easy or hard about the tasks, and about preferences for features to be incorporated into our design for an implemented solution-authoring tool.

The study showed that subjects with no understanding of the underlying technology are both comfortable with the approach and capable of applying it to generate quality solution models. Furthermore, subjects showed that they could generate quality solution models and understand representations of them but certain concepts presented challenges, calling for special attention in tool design. More specific findings are summarized below.

- Subjects understood manually created traces of their demonstrated actions. Furthermore, they found it natural for the default to be that actions are ordered in the trace as demonstrated.
- Our proposed annotations covered the types of generalizations that subjects wanted to express, but some of the distinctions between the annotations were not well understood. In particular, annotations designating options and alternatives were intended for generalizing actions but subjects used them almost exclusively, and also interchangeably, to generalize ingredients and implements instead.
- With minimal instruction, subjects were able to specify annotations to generalize their demonstrations; however, their annotations were not always fully comprehensive. Subjects showed good coverage in specifying alternatives (ingredients, tools, subtask options), identifying optional steps (or groups of actions), and providing acceptable ranges (for quantities).
- Subjects often missed relaxing ordering constraints but responded appropriately when asked whether a later action could be performed before an earlier one. Thus, while subjects were capable of relaxing ordering constraints, this suggests the need for some type of assistance to help improve coverage.
- Subjects wanted an intelligent prompting capability to help them identify missing annotations.

These findings were used to inform the design of the exercise solution editor, described above.

APPLICATION: M4 MAINTENANCE

The SAVE framework is domain independent, enabling it to be applied to a broad range of training tasks that involve procedural skills. Our initial application of the framework has been to the training of weapon maintenance, following the detailed procedures for an M4 series rifle carbine described in Soldier Training Publication 21-1-SMCT (U.S. Army, 2012). We focused specifically on two maintenance tasks: weapon clearing and disassembly.

An existing high-fidelity 3D model of the M4 was obtained for the prototype. To facilitate rapid development, a low-polygon version of the model was created using digital content creation tools to reduce asset load time and to improve rendering performance. Additional adjustments were made, such as setting local coordinates for proper rotation of individual components and flattening of the scene graph hierarchy for easier component manipulation.

The initial S3D annotation file for the M4 was manually drafted and iterated upon as refinements to the 3D model and ontology were made. This XML-encoded file identifies the location of the model and ontology using Uniform Resource Identifiers. Individual markup elements link each 3D node (e.g., magazine, bolt, selector lever, etc.) with a corresponding semantic class defined by the ontology. The S3D file was later extended to include grouping information for dynamic control of 3D geometry objects in defined groups, such as during component disassembly.

The ontologies developed for the M4 domain contain 115 classes, 193 rules, and 18 individuals (e.g., the shooting range, the M4, and its various parts). The class hierarchy contains classes for all relevant components of the M4, and properties that show how the components fit together. The ontology also models actions that can be performed by a user while completing various maintenance operations training exercises (e.g., remove the magazine, lock the bolt open, place the selector lever on safe) as well as actions for the loading and firing of the weapon.

To support reuse across domains, a generic upper ontology was designed that includes common object manipulation actions such as push, pull, insert, extract, and turn; this reusable model was adapted from an existing source (Vujosevic and Ianni, 1997). A mechanics ontology was created for more specific actions such as “change the position of a switch” and “turn a screw,” and their requisite classes “switch” and “screw.” This level of specificity sufficed to model all of the actions required for the clearing and disassembly tasks described in (U.S. Army, 2012).

When an exercise begins, the M4 is displayed in the EUI. The user can apply generic actions (push, pull, etc.) through context-sensitive menus that show allowed operations for each component when selected. When an action is performed, the ontology reasoner determines the effects of that action and updates the world state. The reasoner also provides services to the assessment module to support the evaluation of constraints on objects (e.g., that the M4 selector switch must be set to *Safe* mode). Once the entire exercise is completed to the satisfaction of the learner, an assessment of the performance is conducted and appropriate feedback is provided in the EUI.

RELATED WORK

The Steve system (Rickel and Johnson, 1999) pioneered training assistants for VEs. In this system, a pedagogical agent both teaches procedures for operating electro-mechanical devices and provides assistance to learners as they complete training tasks. While Steve encompasses both interactive critiquing and assistance, our work focuses on post hoc assessment of activities in VEs, with our approach supporting more exploratory forms of assessment rather than being tied to specific solution paths. Our work also provides a framework for defining the semantic models used in assessment, whereas those models are hard-coded in the Steve system.

In recent years, there have been numerous systems built across a range of domains that explore VE-based training. Like Steve, many of these assume predefined semantic models embedded in the VE, although there are exceptions. (Maderer et al., 2013) describes a training architecture that holds semantic information about items in repository external to the VE. The assessment logic is limited to matching against single events rather than complex procedures comprised of action sequence, as presented in our work. (Kessing et al., 2012) demonstrate a game framework in which 2D sprite objects are semantically classified, with authoring tools accessing class libraries stored in a database. The framework is able to reason about basic interactions between objects, including state tracking, however there is no exercise specification or learner assessment.

The use of end-user programming techniques for authoring content for intelligent tutoring systems originated with the work of (Blessing, 1997), who applied it to learn production rules for a cognitive tutoring system. Similarly, (Angros et al., 2002) describe programming-by-demonstration techniques to support acquisition of procedural knowledge for the Steve system, which were supplemented by queries to the demonstrator to determine effects models used for plan generation. While our demonstrations define exercise solutions directly, the usage of demonstration in these systems is to create general problem-solving models from which tutoring actions are then derived. Programming by demonstration is used in (Koedinger et al., 2004) to develop content for example-tracing tutors but requires explicit demonstration of all possible solution paths; in contrast, our assessment module supports generalization through author annotations, enabling a much more compact representation of the solution space for exercises that have large numbers of related solutions. The work in (Mohammed et al., 2005) on training for satellite management is more closely aligned with our approach to solution authoring, consisting of a similar combination of demonstration plus annotation.

CONCLUSION

This paper presents the SAVE framework, which supports training for procedural tasks in virtual environments. Several aspects of this work are noteworthy. First, the assessment capability is explicitly designed to support training of complex tasks with a range of possible solutions. While rooted in an example-tracing methodology, the use of flexible graph matching enables assessments in which learner actions can deviate significantly from allowed solutions, thus enabling exploratory learning during exercise completion. Second, it treats semantic specification as a first-class activity, providing interactive tools for content authors to define semantics for the VE elements that comprise the training environment rather than requiring system developers to hard-code the semantics ahead of time. Third, our novel approach to solution specification through a combination of programming by demonstration and annotations enables subject matter experts to define solution models, rather than requiring technology experts.

The technology described herein, while applied to the M4 maintenance task in the prototype, is domain independent and well suited to a range of training tasks. Virtual environments have broad appeal when real-world practice incurs significant time, expense, or risk. The assessment and solution authoring capabilities are best suited to training tasks where a range of solutions is possible and where the completion of training exercises might involve exploratory actions or variability, rather than adherence to a fixed sequence of actions. While not addressed in the current system, the general approach also provides a basis for evaluating the quality of a solution, e.g., a learner who performed unnecessary actions could be graded to reflect the suboptimality of their actions.

Candidate training targets include the operation of complex artifacts (hardware or software), diagnosis and repair of complex devices, or even synthesis tasks (such as construction). In addition to object-centric training, the approach could also be adapted to individual and small unit collective training in more expansive environments, supporting the learning of tasks such as terrain navigation or movement under direct fire. The semantically enabled tracking and assessment capabilities of this framework could be integrated with operational training simulators, whether they are virtual, constructive, or game-based, potentially extending the training value of existing systems.

A critical element for the success of our approach is the semantic layer that enables the understandability of actions performed by a user in the VE. For the M4 domain, creation of this layer focused primarily on annotating components and groupings, as well as defining allowed actions; while not trivial, these are relatively straightforward to define. Domains for which more complex relationships are essential to assessing performance will present a bigger challenge, as they require the means for content developers to define those relationships in the VE and for the assessment module to reason more broadly about state and contextual effects.

ACKNOWLEDGMENTS

This material is based upon work supported by the Advanced Distributed Learning Initiative (ADL) under Contract No. W911QY-14-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of ADL. The authors thank Daniel Elenius, Chris Jones, Scott Krehbiel, John Pywtorak, Rukman Senanayake, and Michael Wessel for their contributions to the work described in this paper.

REFERENCES

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

Angros, R., Lewis Johnson, W., Rickel, J. & Scholer, A., (2002). Learning domain knowledge for teaching procedural skills. In *Proc. of AAMAS*.

Blessing, S. (1997). A programming by demonstration authoring tool for model-tracing tutors. Intl. Journal of AI in Education, vol. 8, pp. 233–261.

Brawner, K., Holden, H., Goldberg, B., and Sottilare, R. (2012). Recommendations for modern tools to author tutoring systems. *Proc. of I/ITSEC.*

Koedinger, K. R., Aleven, V. Hefferman, N. McLaren, B., & Hockenberry, M. (2004). Opening the door to non-programmers: authoring intelligent tutor behavior by demonstration. In *Proc. of the 7th Annual Intelligent Tutoring Systems Conference*.

Kessing, J., Tutenel, T., & Bidarra, R. (2012). Designing semantic game worlds. In *Proc. of the 3rd Workshop on Procedural Content Generation in Games*.

Maderer, J., Gütl, C., & Al-Smadi, M. (2013). Formative assessment in immersive environments: A semantic approach to automated evaluation of user behavior in Open Wonderland. In *Proc. of the Immersive Education (iED) Summit*, Boston, MA.

Mohammed, J., Sorensen, B., Ong, J., & Li, J. (2005). Rapid authoring of task knowledge for training and performance support. Proc. of I/ITSEC.

Myers, K., Gervasio, M., Jones, C., McIntyre, K., & Keifer, K. (2013). Drill evaluation for training procedural skills. In *Proc. of 16th International Conference on Artificial Intelligence in Education*, 561–570.

Myers, K. & Gervasio, M. (2016). Solution authoring via demonstration and annotation: an empirical study. In *Proc. of the IEEE International Conference on Advanced Learning Technologies.*

Rickel, J. & Johnson, W. L. (1999). Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence, 13*.

U.S. Department of the Army (2012). Maintain an M16 Series Rifle/M4 Series Rifle Carbine. In *Soldier's Manual of Common Tasks, Warrior Skills: Level 1*, Soldier Training Publication No. 21-1-SMCT, PIN: 059832-000, 3-1–3-8.

Vujosevic, R. & Ianni, J. (1997). A taxonomy of motion models for simulation and analysis of maintenance tasks. Tech. Rep., United States Air Force Armstrong Laboratory, AL/HR-TP-1996-0045.