Chapter 27 Tiering, Layering and Bootstrapping for ITS Development

Eric Domeshek, Randy Jensen, and Sowmya Ramachandran
Stottler Henke Associates, Inc.

Introduction

Intelligent tutoring system (ITS) authoring tools aim to lower the cost of code and content development, maintenance, and reuse. We discuss three techniques for cost-containment: *tiering*, *layering*, and *bootstrapping*. Our discussion focuses on the critical task of assessment authoring for *situated tutors* (Schatz, Oakes, Folsom-Kovarik and Dolletski-Lazar, 2011). Situated tutors are a class of ITSs where training is conducted in a scenario-playing experiential environment with intelligent adaptive instruction, including micro-adaptation within scenarios and/or macro-adaptation across scenarios (Shute, 1993). Exercise environments are often quite complex—e.g., simulations of helicopter flight controls, ship battle stations, sensor suites, or command and control systems—and often include components for interacting with simulated teammates, customers, or adversaries. Student assessment is complicated by the nuances of the domain and task, the need to track activity in such complicated simulations, and the need to generate simulation behaviors to create particular learning opportunities. Based entirely on the data and cues available from the training environment, automated assessment mechanisms are responsible for producing judgments of performance at a fidelity that meets training objectives by being sufficiently comparable to human instructor assessments. The complexity of the assessment mechanism in this kind of environment often translates to significant development costs, and thus the need for authoring techniques aimed at reducing costs by structuring the process, component, and content development tasks.

A tiered structure of authoring tools offers a way to tailor knowledge elicitation and engineering for different classes of experts, ranging from those with domain expertise or instructional knowledge, to authors with skills in domain or task modeling, logical and symbolic reasoning, basic scripting, or even advanced programming. A layered approach to modeling allows for composition of model components. It promotes reuse of general knowledge where feasible, while allowing for context-specific knowledge to fill gaps as needed. A bootstrapping approach involves generalizing assessment knowledge from specific instances to scenario-independent mechanisms. Bootstrapping techniques we have applied include incremental rule condition generalization and student action templates created by demonstration and generalization.

These techniques fit an ITS development approach emphasizing incremental example-driven evolution over upfront complete model development. We aim to gain the advantages of rapid/cheap initial capability while still ensuring that instructional unit costs taper over time. We describe our experience building ITS authoring tools that embody approaches to tiering, layering, and bootstrapping.

Related Research

Tiered authoring is an intuitive solution to the challenge of ITS authoring and, not surprisingly, has been implemented in one form or another by a variety of authoring tools. Murray (1999) discusses meta-authoring tools as a potentially effective approach to addressing the usability and power trade-off. Meta-authoring tools are a means for creating special-purpose authoring tools using general-purpose authoring tools. The latter are designed to be applicable to a wide variety of domains and support several types of pedagogical approaches and thus would present a larger degree of authoring complexity. The idea is that

highly skilled authors could use these tools to create special-purpose authoring tools that are targeted at a specific domain and a subset of pedagogical styles (Qiu & Riesbeck, 2005; Hsieh, Halff & Redfield, 1999). Limiting the scope of the tool in this manner makes it possible to design these authoring tools to be more usable and less demanding in terms of authoring skills. Meta-authoring is an example of the tiered authoring approach that we discuss below. For a more recent example, Nye, et al. (2014) describe a tool that uses a tiered approach for augmenting web content with AutoTutor-like dialogues.

Layered authoring of ITS content is primarily intended to enable and promote reuse. This fits one of the authoring tool methods enumerated by Murray (1999). However, given our bias toward example-driven situated tutor development, we focus on reuse across scenarios rather than across entire tutor applications. Layering is *not* aimed at reusing preexisting media or courseware as in REDEEM (Major, Ainsworth & Wood, 1997) or the Shareable Content Object Reference Model (SCORM) standard (ADL, 2009), nor is layering primarily concerned with reusing computational or user interface components (e.g., as in SIMQUEST; deJong et al., 1998). Layering, as presented here, would not make sense in the context of authoring a fully general domain model capable of solving any problem the tutor might pose to a student.

Bootstrapped acquisition of domain knowledge has been gaining traction in recent years. A common approach is to use machine learning algorithms to learn initial domain knowledge and refine it on an ongoing basis (Kumar, Roy, Roberts & Makhoul, 2014; Aleven, McLaren, Sewall & Koedinger, 2006). More recently, SimStudent advances the concept even further where the ITS is an active learner, i.e., it learns from an initial set of demonstrated solutions and refines its knowledge by actively validating it on examples while asking for feedback and help, much like a student. Another bootstrapping approach is limited to using student performance data to improve a tutor's assessment or student modeling knowledge while the initial knowledge itself is handcrafted (Baker, Corbett & Aleven, 2008; Barnes & Stamper, 2008). However, such bootstrapping is primarily envisioned as an automated process, whereas we emphasize the more pragmatic approach of keeping authors in the loop to deal with the commonly required representational shifts.

Discussion

Tiered Authoring

The challenge of ITS authoring lies in developing user-friendly tools that allow subject matter experts or instructional designers to create complex pieces of knowledge. The targeted authors typically do not possess the kinds of computational/logical modeling skill required to create the knowledge that informs ITSs. This skill gap is often too large to be bridged by authoring tools. One way to reduce this gap is to limit the complexity (breadth and/or depth) of knowledge provided to the tutor, thereby reducing modeling complexity. However, this comes at the cost of reducing the "intelligence" of the ITS. An alternative is to partition the space of the knowledge to be authored into sections that can be authored by different people with different skillsets. One approach is to partition the knowledge into modules, each of which might require a different skill set for authoring. For example, an author with expert modeling skills may author performance assessment rules while an instructional designer might configure the pedagogy. An alternative way to partition is to develop tiers of knowledge with one tier combining and specializing another. Templates are an example of intermediary structures or abstractions that can be combined together and instantiated to capture the knowledge required for assessment and tutoring. A tiered approach to authoring provides a way to divide and distribute the task so as to match the skillset and knowledge of the variety of contributors collaborating on the ITS.

The EarthTutor ITS and authoring tool illustrates this approach. This ITS was designed for NASA to teach remote sensing image processing, a domain in which students analyze satellite data using an image

processing application. The objective of EarthTutor is to teach students to use image processing tools by completing exercises related to specific questions about an image. EarthTutor structures an exercise as a series of cards, each containing interactive behaviors embedded in HTML pages. The interactive behaviors consist of questions and real-world tasks the student must complete in the host application. Embedded in these behaviors is the logic for monitoring the student's actions, presenting feedback, and updating the student model.

EarthTutor provides a tiered tool suite for authoring these behaviors. At the foundational level, advanced authors use a graphical flow chart tool to combine ITS and host-application primitives into hierarchies of reusable parameterized assessment behaviors. A novice tier authoring tool, then, allows less skilled authors to use previously defined flow charts from the behavior library to create interactive cards for exercises. The novice tool enables authors to select *behavior templates* from the behavior library, instantiate their parameters, and embed them in a card with other HTML content. Adding an instantiated template to a card indicates (1) that the flow chart linked to the template should be executed when the card is displayed, and (2) that the student interface should replace the template with a user interface (UI) component (defined by the advanced author in the flow chart). This approach allows novice authors to tailor tutoring behaviors to their own pedagogical needs using parameters, but the interface is reduced to what you see is what you get (WYSIWYG) HTML and simple forms. This novice tier tool also lets authors define a hierarchical course structure in which a course contains labs and labs contain cards. The author can set properties for the courses, labs, and cards such as prerequisites and student modeling parameters.

This two-tiered authoring architecture allows subject matter experts to create image processing exercises with automated intelligent tutoring support by piggybacking on the more advanced authors who populate the behavior library. Since the templates are designed to be reusable objects, the work invested in creating them can be amortized over many exercises.

Figure 11 shows the authoring interface for creating behavior templates. In this example, the author has specified the steps for opening a specific image file using the application's menu. Executing this behavior will show the student the necessary steps to find and open a file, monitor their actions, and provide feedback if they open the wrong file.

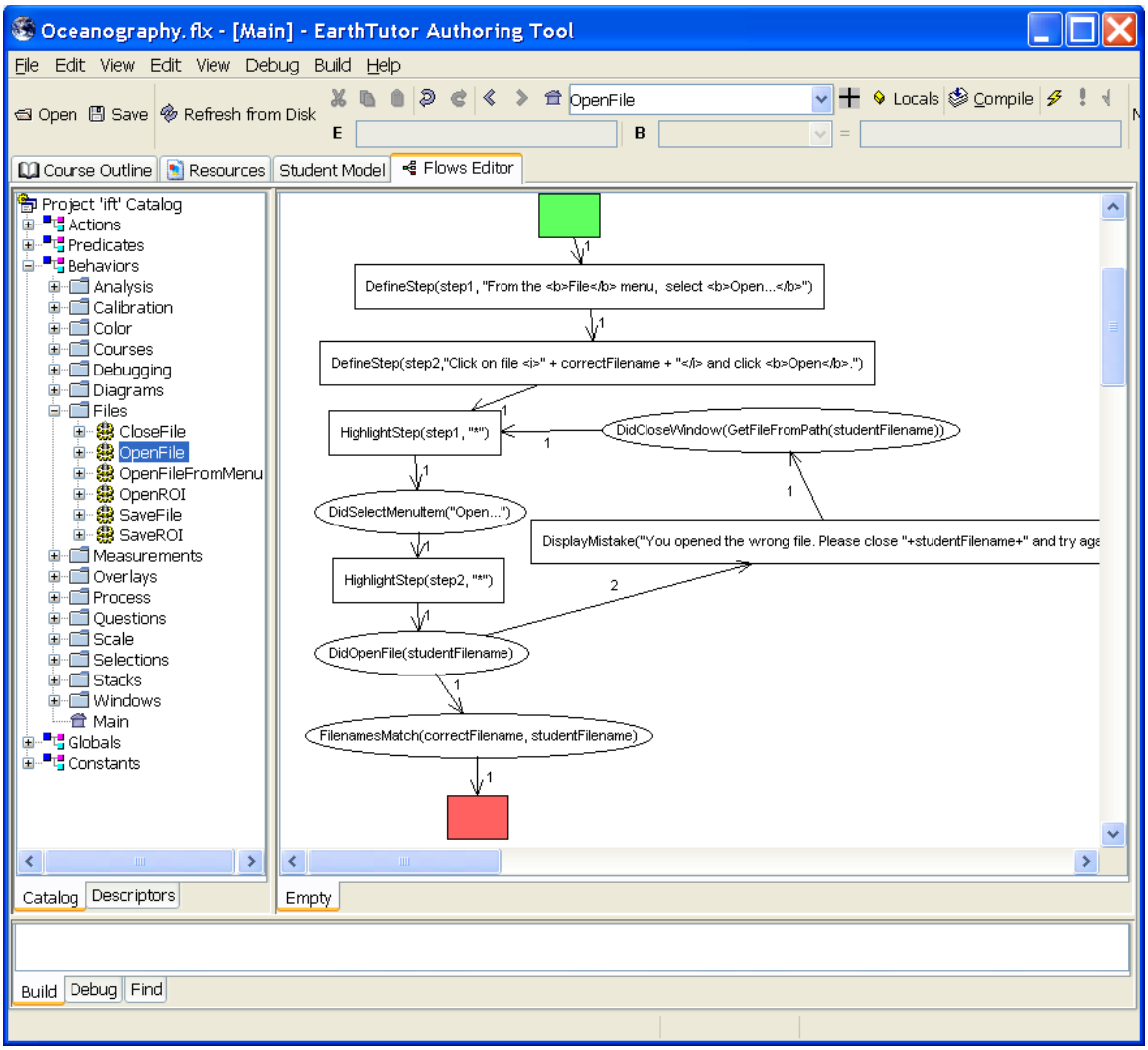

Figure 11. **Flowchart behavior templates are created in EarthTutor's expert authoring interface.**

Figure 12 shows a novice author creating a card for an exercise and embedding previously created behavior templates, including the one for opening a file using the menu. Here, the author has written introductory text and selected two behavior templates, including the one shown above for opening a file. The author has instantiated these templates using simple form-based graphical user interfaces (GUIs). When this card is shown to the student, the templates will be replaced by the GUI that shows the steps for opening a file, and student activity will be monitored as specified in the flowchart above.

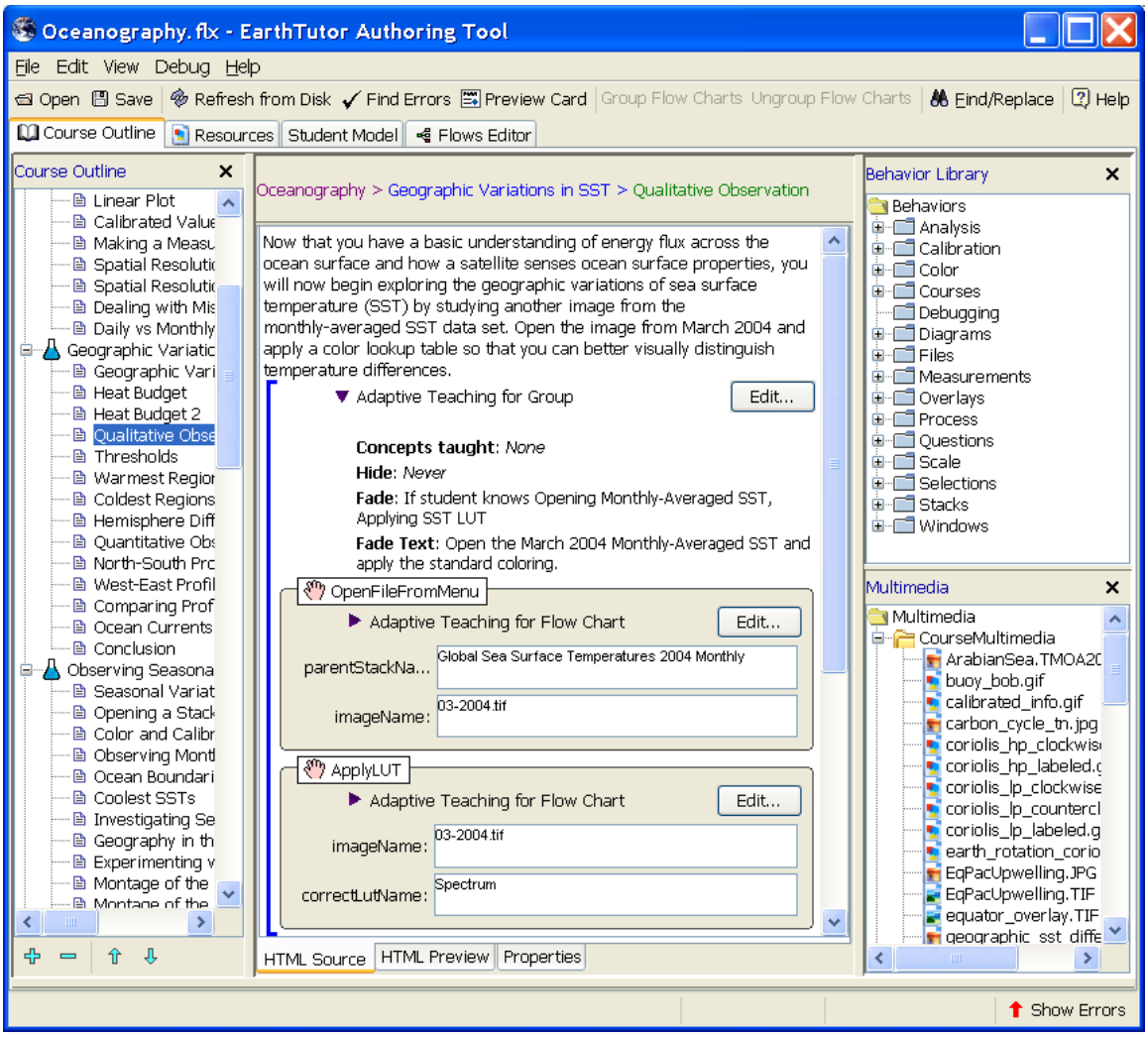

Figure 12. **HTML exercise cards are created in the novice authoring interface by instantiating behavior templates.**

We are using this tiered authoring approach for an ITS being developed to train Navy Information Technology (IT) support staff (ITADS). This is a simulation-based ITS designed to provide hands-on experience with troubleshooting skills and maintenance procedures. The knowledge required for automated assessment of performance—especially for troubleshooting exercises—requires complex modeling and will be constructed by developers or very advanced authors. Once the assessment knowledge has been modeled, less advanced authors will create scenarios that reference this model and tie to frozen sets of virtual machines supporting the simulation. Using simple form-based editors, novice authors can edit student-visible text associated with the scenario and with the model-linked coaching; they can also copy and adapt expert-developed scenarios. The ITS is also designed to use Socratic dialogues as a pedagogical strategy for coaching students. We take a tiered approach to authoring these dialogues as well. Advanced authors use a dialogue authoring tool to create a variety of dialogue structures. Novice authors copy and modify dialogues using form-based editors.

Layered Authoring

Tiering primarily addresses the provision of different authoring tools for different kinds of content most efficiently created by different kinds of authors. The prototypical approach of templating allows a higher volume of content to be generated more quickly by lower-skilled authors, guided and constrained by patterns established by higher-skill (and more expensive) authors. Layering, in contrast, focuses on picking apart a single kind of content (or at least a single view of content that may be composited from related elements) into pieces that have different scopes of applicability or levels of generality. The object is to achieve more reuse of authored content.

Our prototypical example of layering comes from a system that entwines *simulation*, *assessment*, and potential tutor *interventions* into a composite authoring view of linked content. Our Medical Emergency Team Tutored Learning Environment (METTLE) ITS teaches diagnosis, emergency response, and task-specific coordination appropriate for responding to chemical, biological, and radiological attacks. The target trainee is an emergency room physician. In a scenario based on an anthrax attack, the doctor is coached through an initial diagnostic session with a mystery patient in their emergency room (ER). The web-based system supports text-based diagnostic interviewing, media-based physical examination, and form-based ordering of tests and treatments. A cast of other characters can be consulted or may intervene during the scenario, including an ER nurse, a hospital administrator, and staff members at ERs of other nearby hospitals. The tutor provides proactive and reactive hints and feedback, and can also carry out extended Socratic dialogues to review diagnostic logic.

METTLE adopts a theater metaphor in which an exercise scenario is viewed as a sort of dynamic play. A METTLE scenario has a cast, each member of which is assigned a set of behaviors that we think of as "lines" in a nonlinear "script," to be used when triggered by student activity or other scenario events. These behaviors can vary across the scenes of the scenario and based on the state of the character. Lines, then, can be assigned by role, scenario, scene, or state, (or, in the most flexible case, based on some combination of those factors). Lines can include (1) a cue (trigger conditions), (2) a response (the character's scripted actions), (3) side-effects on scenario or character state, and (4) contextual tutor evaluations and comments (including hints, prompts, and feedback).

METTLE allows for composition of scripts and even individual script lines from different sources. For instance, a set of default behaviors can be defined that apply to any character in any situation (e.g., how to handle greetings, farewells, and small talk), while a more specific set of behaviors can be defined for some particular class of simulated characters (e.g., how any character assigned the "patient" role should respond to diagnostic questions). For the patient role, we defined a basic set of several hundred default script lines, providing a reusable set of named rules with cues and "normal" responses covering many standard diagnostic interview questions, examination actions, diagnostic tests, and so on.

When scripting any particular patient for any particular scenario, a subset of these default rules can be extended with situationally important responses, state changes, and tutoring. For instance, a patient with anthrax really only differs from a normal healthy adult on a small set of key diagnostic indicators. Authors can compose scenario-appropriate diagnostic question/answer script lines by taking the trigger from the role-general form of the behavior (the question stays the same), while overriding the response to fit the scenario (the answer is tuned to fit the results that would be expected for an anthrax patient). Entirely new rules can be added for behaviors that only make sense in the context of the scenario (or some scene or character state). For instance, if an important aspect of the case is how the patient got the disease, then a back-story can be introduced with a set of custom question and answer behaviors bearing on their recent activities, who they were with, and how those people are faring.

Consider a pair of example behaviors used in the anthrax scenario. First, there is a standard diagnostic interview question—appropriate in cases where the underlying issue might be an infectious disease—that checks if anyone else the patient knows is suffering from a similar problem. In the generic patient script there is a line named **Complaint-Others** that is specified with cues such as *"Do you know anyone else with the same symptoms?"* and a default answer of *"No."* In the anthrax scenario, this is an important question and so additions and modifications are layered onto the default behavior. For starters, its answer is overridden so that the patient says: *"Yeah, my cousin John has come down with some fluey thing since we last saw each other. My wife says his wife took him to Memorial Hospital today."* The triggering of this behavior is tied to a curriculum point labeled **ED-Diagnosis-Infection** and in the anthrax scenario a proactive prompt is associated to be used by the tutor if this behavior has not been triggered 5 minutes into the scenario: *"You might consider asking whether Ryan knows anyone else who has what he has."* Our second example is a totally new script line introduced for this scenario. Once it is revealed that the patient's cousin is also sick, there should be a follow-up line of questioning about the cousin. Accordingly, this scenario introduces a new line named **Others-Cousin-When** with cues that include *"When did you last see your cousin?"* eliciting the answer *"We went to a basketball game together with another friend of mine maybe 5 or 6 days ago."*

These examples illustrate composition of aggregate scripts from behaviors defined at different layers such as for generic characters, generic patients, and some particular patient in a scenario. They also illustrate composition of individual script lines from fragments defined at different layers, such as a question/answer behavior defined for generic patients being overridden with an answer appropriate to a particular patient and associated tutoring interventions.

METTLE behaviors are composed from an extensible application-specific rule condition/action language. Extensions to that language can be viewed as an expert level authoring tier similar to EarthTutor's advanced authoring of behavior templates. In addition, it might turn out that different tools are appropriate for different layers, or that different classes of authors are best suited to providing different layers of content. Nonetheless, when it comes to building up behaviors in layers, the primary issue is not division of labor, but rather content reusability—across exercises, courses, and possibly even domains.

Bootstrapped Assessment

Bootstrapped authoring, as applied to automated assessment, is an incremental development process where the cost of authoring is reduced with successive spirals or releases. Starting with example-based scenario-specific content and training mechanisms, authors incrementally generalize to create successively more reusable components. Each iteration offers cost savings over the last, coupled with wider reusability for the next.

Before proceeding to examples, we explicate what we mean by generalized assessment mechanisms in the context of a situated tutor. A common tradeoff in designing automated assessment is the choice between an example-based or model-based approach. Example-based assessment makes inferences from the case-specific conditions that apply in a particular scenario, without regard for how the same concepts would appear or be assessed in other scenarios. Because example-based assessment mechanisms can be essentially hard-coded with unique knowledge associated with a specific scenario, learner, or context, they are often easy to rapidly prototype. This can be very productive for the early stages of development when requirements are still being refined. However, as the number of scenarios grows, the example-based approach must be essentially replicated for each new scenario.

In contrast, a model-based approach seeks to capture more general knowledge that reduces the cost of authoring new scenarios. In the broadest sense, an assessment model attempts to represent knowledge,

skills and aptitudes (KSAs) and how they're applied in scenarios, without relying on unique scenario-specific knowledge. In practice, there are numerous approaches to model-based assessment, ranging from those that represent recurring but recognizable constraints on good performance, to those that aim to represent a comprehensive space of possible actions together with the underlying cognitive states that produce those actions. Regardless of the precise formulation, we emphasize the goal of *scenario-independence*. On one hand, the effort required to achieve scenario-independent assessment doesn't easily scale *down* to the early development stages where prototyping is useful. So in the short term of initial prototyping, a purely model-based approach is inherently more costly and time-consuming to implement than a purely example-based approach. However, in the longer term, a generalized assessment model reaps authoring cost benefits precisely because of the scenario-independence. A generalized model can also theoretically be abstracted further, to shed the specific constraints of a given simulation or exercise environment, and yield cost savings for transitions to other platforms.

Given the practical benefits of example-based methods in the short run and model-based methods in the long run, the bootstrapping approach seeks a transition from the former to the latter in the course of assessment authoring. This combines the expedient of an example-based approach for early development, with the future authoring benefits and cost savings associated with a generalized model. The concept of bootstrapped content authoring can be applied over successive development spirals of a scenario-based ITS, in tandem with expansions in either or both the collection of scenarios or the core ITS assessment capabilities.

This bootstrapping approach was employed in the development of a game-based trainer for the Army's US Military Academy (USMA) at West Point, called Intelligent Game-based Evaluation and Review (InGEAR). InGEAR is integrated with a tactical decision-making game called Follow Me, which is used at West Point to teach small unit leader tactics in dynamic, experiential scenarios. The project objective was to extend the reach of instructors and allow self-directed learning for cadets using the game environment. Before InGEAR was developed, Follow Me was used entirely with facilitated classroom learning, where all performance assessment and feedback in exercises was the province of human instructors. The USMA instructional staff designed an existing set of scenarios to exercise tactical concepts with varying degrees of difficulty, and assessed cadets' performance by applying accumulated knowledge of scenario dynamics. For InGEAR, this existing scenario knowledge provided an excellent baseline for a rapid prototyping effort in the first spiral of development. Example-based assessments were developed within 4 months of the project start, following the lead of established instructional knowledge.

One of the benefits of rapid prototyping in this manner is that it produces a useable training capability early on. However, with InGEAR the objective was to deliver scenario-independent mechanisms that could assess the same tactical concepts when relevant in future scenarios to be created or modified by the USMA instructional staff after the InGEAR development effort. The combination of short-term prototyping goals and long-term project goals motivated a bootstrapping evolution from an initial set of example-based assessments to an eventual set of generalized scenario-independent assessments using a constraint-based model.

An example of this evolution involves the assessment of cover and concealment in tactical movement. Initially with existing Follow Me scenarios used at West Point, instructors were so intimately familiar with the terrain and enemy positions that they could immediately point to good and bad areas of cover and concealment. Following this lead, the initial example-based assessments in the first spiral used scenario-specific annotations to score areas of terrain, applying a figure of merit for the quality of cover and concealment in significant areas. This was easy to develop quickly, and it provided a sample working assessment to review with instructors (along with automated feedback and other capabilities). It also served as an effective primer for the development team to quickly gain an understanding of the domain, which facilitated the ongoing collaboration with both the USMA staff and the developer of Follow Me.

However, this example-driven approach could not be easily extended to future scenarios, so the next spiral required a more general model-based approach to assessing cover and concealment.

In order to develop a scenario-independent assessment for cover and concealment, the methodology was to review existing scenarios where the tactical principles were applied, and to abstract the key concepts across settings. From that, a mechanism could be constructed to reason about the merits of a tactical position with respect to those concepts, in any given scenario. The key concepts in this case involve visibilities in relation to actual or likely enemy positions, and visibilities in specific terrain (e.g., the inherent level of exposure on a ridgeline versus a wooded area). For this application, the game environment already dynamically calculates visibilities between units and between terrain positions. The screenshot in Figure 13 shows an example of terrain visibility in the game (represented as a pixelated overlay) from the position of a particular machine gun unit (also shown with its sector of fire as a wedge shaped graphic).

Figure 13. **The Follow Me game shows machine gun section visibilities and sectors of fire.**

During an exercise, instances of detection by enemy units trigger game notifications, contributing to half of the generalized assessment for cover and concealment. However, it is more complex to implement a real-time assessment of the quality of a position in terms of terrain exposure. To support such assessment, we constructed an authoring utility to pre-process the terrain database for any given scenario by generating exposure scores for all positions (represented as terrain tiles). These scores can then be used during execution for real-time assessment, without requiring heavy processing during the exercise and without requiring explicit manual instructor annotation of the terrain in authoring. The resulting generalized assessment for cover and concealment was scenario-independent, with minimal requirements on authors seeking to activate this assessment for a new scenario. From a methodological standpoint, the implementation benefited from the earlier knowledge acquired with the example-based implementation, which accelerated the development of the subsequent more general mechanism. As a further benefit, the general assessment's performance could be compared with the earlier example-based versions as well.

For each scenario-independent assessment implemented for InGEAR, the final step to support authoring was to produce a specification for the parameters required to configure and apply the assessment mechanism in a scenario. In some cases, the parameters are thresholds for time, distances, survivability, or other factors that instructors determine will delineate performance standards (such as pass/fail). In other cases, the parameters involve a simple specification of a game artifact, such as an objective area to be secured as part of a tactical task.

Recommendations and Future Research

The three approaches discussed in this chapter, tiering, layering, and bootstrapping, hold promise for addressing the trade-off of power vs. usability in the design of authoring tools, while enabling cost-savings through content reuse and restructuring. Further research is required to build such tools and validate them for a variety of ITSs. The Generalized Intelligent Framework for Tutoring (GIFT) can facilitate this research by providing a unified framework for collaboration on this research.

GIFT provides a decomposition of typical ITS functionality that aligns well with a range of applications and capabilities. For instance, the architecture presented by Ragusa, Hoffman, and Leonard (2013) has many broad correspondences to the architecture of the ITADS system mentioned earlier: both separate management/monitor functions from the tutor user interface, which is separate from any simulation/game modules (which are linked to the ITS through an interface module); both have user management and learning management modules; and both have domain, learner, and pedagogy modules.

Domain knowledge—specifically performance assessment rules—can be specified in the GIFT framework within extensible markup language (XML) domain knowledge files (DKFs). GIFT provides a Domain Knowledge File Authoring Tool (DAT), an XML editing tool for creating and editing these rules. The DKF—and its associated DAT—provide a means to define *assessments* and *state transitions*. **Assessments** use a hierarchy of *tasks*, *concepts*, and *conditions* to cover *runtime performance* assessment (during exercises) and *scoring rules* (aggregate after-exercise scores). **State transitions** itemize *changes* in learner state that are of interest (including, of course, assessed performance states), each with a list of *strategies* the tutor might use to respond to those changes.

Within GIFT's general module breakdown and domain modeling framework, we see several possible extensions that might support tiering, layering, and bootstrapping.

An obvious way to support tiered authoring within this framework is to allow parameterized rules and create a GUI-based authoring tool in addition to the DAT for novice authors to instantiate parameters. Another useful capability would be to support multiple simultaneous authors so that the task of rule creation can be distributed more fluidly. With these changes, expert authors could create complex logic while novice authors could create simpler rules. This capability should be supported by associated integration and testing tools for the overall set of rules. A more advanced approach might be to provide the capability to create flowcharts representing branching sequences of assessments and state transitions (e.g., to represent procedural tasks). An expert tier authoring tool could be developed for creating such flowcharts as a part of a DKF specification, while a novice tier authoring tool supported selection and instantiation of templates.

Layered authoring, as exemplified in METTLE, could also be introduced into the GIFT framework. One challenge here is our example's relatively tight coupling between simulation/game construction, assessment authoring, and tutor intervention specification. However, if it is most natural for a scenario-focused author to think about exercise behavior, performance evaluation, and coaching in tandem then authoring tools should provide a view that couples those structures, even though an underlying

architecture might divide the simulation/game from the assessment engine from the tutor utterances. At the same time, the tools should provide a view that helps authors understand the contextually composited form of a behavior or rule, even though it combines new and reused pieces from different scopes. Again, this view should be available irrespective of how the generic underlying ITS architecture wants to divide up the included, reused, and overridden bits of knowledge.

Bootstrapped assessment authoring may also be facilitated with the GIFT framework, by adding structure for regression testing, to be integrated with the analysis testbed methodology. Naturally if assessment mechanisms will undergo an evolution as they are incrementally generalized, then some form of regression testing is desirable to verify that the assessment results from a generalized mechanism match those produced from earlier example-based assessments in a battery of specific scenarios. The GIFT framework may be an effective place to introduce such testing artifacts, because its domain module is designed to consume assessment outputs from an instrumented exercise environment, while being abstracted from the internals of the implementation in the environment. This inherently supports the abstraction between the GIFT domain module and pedagogical module. This same abstraction is relevant to a potential additional function for the GIFT analysis testbed methodology, which seeks to refine and validate learning outcomes in different conditions, such as an authored tutor versus traditional classroom learning. This comparison methodology would be useful for validating an evolving assessment approach developed in a bootstrapping fashion—to compare an initial baseline of example-based assessment mechanisms to subsequent more generalized iterations or spirals

References

ADL (2009). SCORM® 2004 4th Edition Content Aggregation Model (CAM) Version 1.1. *Advanced Distributed Learning (ADL).* Retrieved from: http://www.adlnet.gov/scorm/scorm-2004-4th/.

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.

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.

de Jong, T., van Joolingen, W.R., Swaak, J., Veermans K., Limbach R., King S., and Gureghian D. (1998). Combining human and machine expertise for self-directed learning in simulation-based discovery environments. *Journal of Computer Assisted Learning*, **14**(3), 235-246.

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

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.

MacLellan, C. J., Koedinger, K. R. & Matsuda, N. (2014). Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. *Intelligent Tutoring Systems Lecture Notes in Computer Science, Vol. 8474*, 551-560.

Major, N., Ainsworth, S., and Wood, D. (1997). REDEEM: Exploiting symbiosis between psychology and authoring environments. *International J. of Artificial Intelligence in Education*. **8**(3-4), 317-340.

Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal 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*.

Schatz, S., Oakes, C., Folsom-Kovarik, J. T. & Dolletski-Lazar, R. (2012). ITS + SBT: A review of operational situated tutors. *Military Psychology* **24**(2), 166-193.
Shute, V. J. (1993). A macroadaptive approach to tutoring. *Journal of Artificial Intelligence in Education*, **4(**1), 61-93.