Developing Authoring Tools for Simulation-Based Intelligent Tutoring Systems: Lessons Learned

James E. McCarthy[1](✉), Justin Kennedy[2], Jonathan Grant[2], and Mike Bailey[2]

[1] Sonalysts, Inc., Fairborn, OH 45324, USA
mccarthy@sonalysts.com
[2] Sonalysts, Inc., Waterford, CT 06385, USA
{jkennedy,jgrant,mbailey}@sonalysts.com

Abstract. Intelligent tutoring systems have a long history of significantly improving student performance. Unfortunately, they also have a long history of being very expensive to develop while producing very short shelf lives. To address these deficiencies, we developed and evaluated an authoring suite known as the Rapid Adaptive Coaching Environment (RACE). While developing RACE, we frequently encountered tension between maximizing the power of the intelligent tutoring system that authors would produce and minimizing the level of effort associated with producing this tool. In this paper, we will attempt to highlight this tension as we consider the five core design challenges that we addressed in the RACE prototype. We will review the formal and informal evaluations conducted by the development team and discuss how those evaluations led to the maturation of the design.

Keywords: Intelligent tutoring · Authoring systems · User-centered development

1 Introduction

Over the past 30 years, the United States Department of Defense (DoD) has been a leading developer of intelligent tutoring systems (ITSs, Fletcher 1988; McCarthy, 2008). Perhaps one of the most thoroughly researched intelligent tutoring systems in the military domain is SHERLOCK, developed near the beginning of the 1990s (Katz and Lesgold 1993; Lesgold et al. 1992). Like many other military ITSs, SHERLOCK was developed to improve troubleshooting skills, this time in conjunction with the F-15 Avionics Test Station. The original SHERLOCK research demonstrated that about 25 h of practice in that environment had an impact on post-test performance equal to about 4 years of on-the-job experience (Corbett et al. 1997). Gott et al. (1995) followed up with a report that reviewed five separate evaluations of SHERLOCK. That review showed a performance improvement of the 50th percentile students to the 85 percentile (effect size of d = 1.05 standard deviations; an effect considered large and robust).

© Springer Nature Switzerland AG 2019

However, despite this long history and many demonstrable benefits, very few intelligent tutoring systems are currently in use within the DoD. There are several reasons for this, including the characteristics of the tutors that have been developed, the level of effort associated with their development, and the difficulty and cost associated with maintaining these systems over time. The current effort was designed to address these limitations. Specifically, our team set out to develop an authoring system that would make it easier, less expensive, and less time-consuming to develop and maintain intelligent tutoring systems. Our charter mandated that we address problems and systems representative of those encountered throughout the military, and that we do so in a way that was independent of a particular simulation environment or tutoring engine.

The observation that intelligent tutoring systems are very powerful but are hampered by high development and maintenance costs is not a new one. Several "workarounds" have attempted to make intelligent tutoring systems more cost-effective. While many of the approaches have real merit, they all exhibit deficiencies. For example, machine learning approaches have been used to develop a multidimensional representation of what a correct solution looks like within a complex problem space. A student's performance is compared to this standard and a proficiency assessment is made. A critical deficiency of machine learning approaches is that they yield only "black box" models of expertise. The student's score indicates how much his/her performance differs from what the tutor had been taught to expect, but not why the performance is different. Real-time feedback is not possible.

An alternative to machine learning approaches is to use formal cognitive modeling architectures and languages to develop glass-box cognitive models of expert performance. These architectures generally provide a programming language that allows users to model human cognitive processes and behaviors. However, these languages almost always are specified at a primitive level, similar to assembly code (Cohen et al. 2005; Ritter and Koedinger 1995). This makes it very time consuming to develop the models and usually the models are over-specified. That is, they include details that either cannot be supported or that are irrelevant to the task at hand (Amant et al. 2005). To address these concerns, several groups have worked to simplify this programming process. Across these efforts, the strengths and weaknesses of the high-level language approach very nearly complement those of the machine learning approach. High-level languages can produce glass-box representations of expertise and offer the opportunity to provide real-time performance assessment and feedback. Unfortunately, they still are based on relatively conventional task analyses and require manual model development involving a relatively large team of domain experts, knowledge engineers, and software engineers. While this development process is unquestionably easier, it still requires technical knowledge of both the language itself and the underlying architecture(s). As a result, while high-level languages can reduce development and maintenance costs, they do so incrementally. They do not provide the order-of-magnitude cost savings we desire from this project.

Instead of adopting either of these approaches, our team focused on demonstration-based authoring of intelligent tutoring systems (cf., Koedinger et al. 2003). We felt that, like machine learning approaches, demonstration-based authoring would allow experts to do what they do best – perform – while also producing glass-box representations like those offered by expert models defined with high-level languages. Such a technique would reduce the cost and level of effort associated with expert model development dramatically.

Academic research has indicated that demonstration-based tutors have the potential to increase the efficiency of tutor production by an order-of-magnitude (Koedinger et al. 2003; Aleven et al. 2009). Further, they produce glass-box tutors that can provide real-time feedback and hints. In doing so, they avoid the cost of complex task analyses and replace programmers and instructional designers with subject-matter experts (SMEs). However, this approach is not without problems of its own. These tutors generally have been developed for relatively simple domains such as introductory mathematics. These simple, static domains simplify the problem considerably along a number of dimensions.

One characteristic of the demonstration-based tutors developed for simple domains is that their user interface requirements are easily met. Few user controls are needed and those that are needed are common. As a result, toolboxes of reusable and instrumented widgets are easier to create. Across military domains (aircraft operations, tactical system employment, *etc.*) interfaces are likely to be much more complex, thus building an interface from a toolbox is less practical. Similarly, the simple domains associated with demonstration-based tutors developed so far have been static. Once a problem is presented, it remains the same except for the actions of the operator. This is not true of many military domains; dynamically shifting situations and priorities require the system to constantly reassess which actions are most appropriate. Moreover, the content associated with these domains rarely changes in any significant way over the course of years. Across military domains, change is relatively constant.

Within this project, we set to address these challenges within a prototype authoring tool known as the Rapid Adaptive Coaching Environment (RACE). While developing RACE, we frequently needed to balance our efforts to maximize the power of the intelligent tutoring system that authors would produce while minimizing the level of effort associated with producing the tool. In the discussion that follows, we will attempt to highlight this tension as we consider the five core design challenges that we addressed in the RACE prototype:

1. Assessment of Evolving Situations
2. Capturing State and Action Information from Arbitrary Simulations
3. Exporting Assessment Models for Arbitrary Intelligent Tutors
4. Specifying Evaluation Standards, and
5. Providing Instructive Hints and Feedback.

2 Baseline Approach

Our initial approach to developing RACE focused on the development of a learning objective hierarchy and the association of each LO with a scenario event that would serve as a practice opportunity. The following subsections provide an overview of this process.

2.1 Outlining

The grand vision for RACE included the ability to develop both simulation-based intelligent tutoring that would allow students to independently practice skills and media-based adaptive interactive multimedia instruction (IMI) that would allow students to master the fundamentals of a given domain. As such, it was very important to the development team to establish a solid foundation for this work through the development of a well-structured hierarchy of learning objectives (LOs).

However, we were also aware that the likely users of RACE were not instructional designers *per se*. Rather, they were likely to be SMEs who were, perhaps, serving as instructors. Therefore, we felt that it was important to scaffold their performance. To do this, we started by asking authors to identify the thing that was probably most familiar to them – the tasks that they wanted their students to be able to perform when they were done with training. The development team felt (and later evaluations confirmed) that the SMEs/authors would have clear expectations about this.

Next, we asked the authors to take one small step away from tasks and think about what the students would have to do to complete each task. That is, what skills are required for successful task performance? Scaffolds, in the form of general directions, more detailed explanation, and access to a very detailed "how-to" guide supported authors in this process.

Just as we asked the authors to identify the skills that were needed to perform the task, we also asked the authors to identify the knowledge that was required to perform each skill. That is, what facts, concepts, principles, *etc.* must be mastered before the student can successfully perform the skill?

The last step in this outlining process guided the authors through defining learning objectives for each of the skill and knowledge elements that they had defined. Within this process, the authors selected (1) an LO Type (Fact, Procedure, *etc.*) that was consistent with the current element type (knowledge *vs.* skill), (2) an LO Level that indicated the depth of mastery that was required (Remember, Use, *etc.*), and (3) a verb that was appropriate for that Type and Level (State, Configure, *etc.*). RACE then asked the author to specify an object for the verb (*e.g.,* "the primary components of the radar receiver" or "the radar receiver for in-port operations"). Together, these elements allowed RACE to define LOs with known outcome types. The final LO took the form "The student will be able to VERB OBJECT" (*e.g.,*"The student will be able to state the primary components of the radar receiver.").

2.2 Developing Practice Scenarios

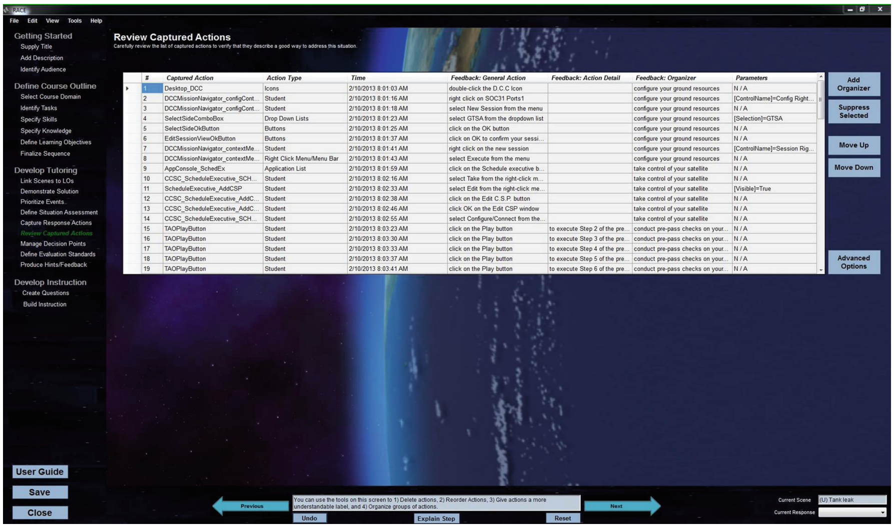
Next, the author was asked to develop practice scenarios. This was a two-step process. First, RACE guided the author through the process of associating each skill-based LO with one or more scenario events that could provide an opportunity to practice the associated skill (or check for mastery of that skill).

The second step in the process was to actually develop a scenario that included that event. RACE was designed to work with any conformant simulation system. As a result, it did not include scenario development support; the development team felt that it was more appropriate to leave that on the simulation "side of the fence."

2.3 Demonstrating Solutions

To demonstrate a solution, the author would select a skill-based learning objective and launch an associated scenario within the simulation environment. For our prototype, we used various applications developed within the Standard Space Trainer (SST) framework (Nullmeyer and Bennett 2008). Both RACE and the simulation connected to a piece of middleware. Using a well-defined interface, the middleware component captured data from the simulation, reformatted it, and passed it to RACE for processing.

To demonstrate the correct procedure for the event (and, by extension, the LO), the author had to do two things. First, the author had to use a RACE interface to define a situation assessment rule that the system would use to recognize that the preconditions for the LO have been satisfied and the student should begin the associated procedure. Second, the author had to demonstrate the appropriate procedure. As the author conducted the demonstration, the simulation passed the recorded actions to the middleware component and on to RACE where it appeared on a display that the author could review (see Fig. 1).



Review Captured Actions

Carefully review the list of captured actions to verify that they describe a good way to address this situation.

#	Captured Action	Action Type	Time	Feedback: General Action	Feedback: Action Detail	Feedback: Organizer	Parameters
1	Desktop_DCC	Icons	2/10/2013 8:01:03 AM	double-click the D.C.C Icon		configure your ground resources	N / A
2	DCCMissionNavigator_configCont...	Student	2/10/2013 8:01:16 AM	right click on SOC31 Ports1		configure your ground resources	[ControlName]=Config Right...
3	DCCMissionNavigator_configCont...	Student	2/10/2013 8:01:18 AM	select New Session from the menu		configure your ground resources	N / A
4	SelectSideComboBox	Drop Down Lists	2/10/2013 8:01:23 AM	select GTSA from the dropdown list		configure your ground resources	[Selection]=GTSA
5	SelectSideOkButton	Buttons	2/10/2013 8:01:25 AM	click on the OK button		configure your ground resources	N / A
6	EditSessionViewOkButton	Buttons	2/10/2013 8:01:37 AM	click on OK to confirm your sessi...		configure your ground resources	N / A
7	DCCMissionNavigator_contextMe...	Student	2/10/2013 8:01:41 AM	right click on the new session		configure your ground resources	[ControlName]=Session Rig...
8	DCCMissionNavigator_contextMe...	Right Click Menu/Menu Bar	2/10/2013 8:01:43 AM	select Execute from the menu		configure your ground resources	N / A
9	AppConsole_SchedEx	Application List	2/10/2013 8:01:59 AM	click on the Schedule executive b...		take control of your satellite	N / A
10	CCSC_ScheduleExecutive_SCH...	Student	2/10/2013 8:02:16 AM	select Take from the right-click m...		take control of your satellite	N / A
11	ScheduleExecutive_AddCSP	Student	2/10/2013 8:02:33 AM	select Edit from the right-click me...		take control of your satellite	[Visible]=True
12	CCSC_ScheduleExecutive_AddC...	Student	2/10/2013 8:02:38 AM	click on the Edit C.S.P. button		take control of your satellite	N / A
13	CCSC_ScheduleExecutive_AddC...	Student	2/10/2013 8:02:46 AM	click OK on the Edit CSP window		take control of your satellite	N / A
14	CCSC_ScheduleExecutive_SCH...	Student	2/10/2013 8:02:55 AM	select Configure/Connect from the...		take control of your satellite	N / A
15	TAOPlayButton	Student	2/10/2013 8:03:23 AM	click on the Play button	to execute Step 2 of the pre...	conduct pre-pass checks on your...	N / A
16	TAOPlayButton	Student	2/10/2013 8:03:30 AM	click on the Play button	to execute Step 3 of the pre...	conduct pre-pass checks on your...	N / A
17	TAOPlayButton	Student	2/10/2013 8:03:33 AM	click on the Play button	to execute Step 4 of the pre...	conduct pre-pass checks on your...	N / A
18	TAOPlayButton	Student	2/10/2013 8:03:37 AM	click on the Play button	to execute Step 5 of the pre...	conduct pre-pass checks on your...	N / A
19	TAOPlayButton	Student	2/10/2013 8:03:41 AM	click on the Play button	to execute Step 6 of the pre...	conduct pre-pass checks on your...	N / A

You can use the tools on this screen to 1) Delete actions, 2) Reorder Actions, 3) Give actions a more understandable label, and 4) Organize groups of actions.

Current Scene: (U) Tank leak

Current Response:

Fig. 1. RACE review captured action screen. Authors use this screen to review, edit, and organize procedures demonstrated within the associated simulation system.

Using this interface, authors could delete or re-order steps to correct minor errors made during the demonstration process. In addition, they were encouraged to organize steps in the process according to the goals that were being pursued. These organization labels were used in the feedback development process.

2.4 Setting Evaluation Standards

After reviewing and refining their demonstrations, authors were asked to define evaluation standards that would specify how precisely they expected students to mirror their demonstrated procedures. Authors used the RACE interface to specify standards for accuracy and pacing. Accuracy standards included things like using the right control or entering the correct value. Pacing standards included the order in which users performed tasks and how much time elapsed between actions.

Different types of actions had different standards associated with them and the authors could choose to enable/disable standards or to adjust the values against which student performance would be judged (including adjusting the "margin of error" that is acceptable.

2.5 Validating Coaching Statements

RACE automatically generated hints and feedback that the intelligent tutoring system could deliver to students to enhance their performance. RACE applies a grammar to captured actions to create the feedback. Each action is associated with three levels of progressively more directive hints. Similarly, RACE associated each correct action with three levels of progressively more detailed feedback. RACE also associated each action with an arbitrary number of coaching statements that the intelligent tutor could deliver when the student made an error. The specific evaluation standards that the author selected were associated with specific types of errors. Each type of error was associated with up to five levels of progressively more detailed feedback. The authors' task was to review this automatically generated feedback and make any adjustments that they felt improved its quality.

2.6 Supporting Multiple Tutoring Environments

Just as RACE was designed to be independent of a given simulation environment, it was also designed to be independent of a given tutoring engine. To pursue this goal, the development team adopted an industry standard for the specification of task models: The W3C XML Task Model (2014). The W3C XML task model includes a meta-model expressed in Unified Modeling Language (UML) and an eXtensible Markup Language (XML) Schema. While the task model has been targeted as the basis for interchange of task models between different user interface development tools, it is general enough to express most of the concepts involved in the RACE assessment model, and can be extended to cover the rest. A key attribute of the W3C Task Model is that it already includes a number of operators to describe relations between tasks. Those operators include interleaving, order independence, synchronization, parallelism, choice, disabling, suspend-resume, enabling, iteration and optional.

For our prototyping effort, middleware was developed that allows Sonalysts' intelligent tutoring system, ExpertTrain (McCarthy 2008), to ingest the W3C Task model and convert the data that it contained into the data structures needed by ExpertTrain to support automated performance assessment and coaching.

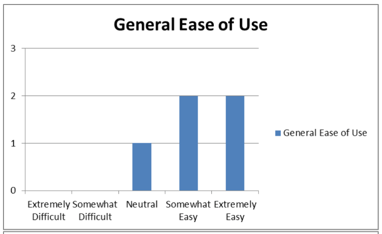
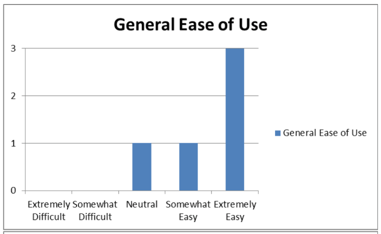
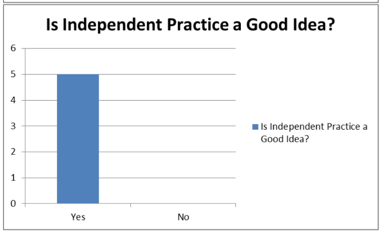
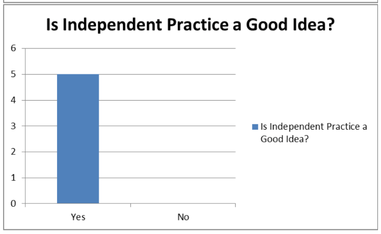
3 Evaluations and Revisions

Throughout the development process, the RACE team conducted a series of evaluations of the emerging prototype. Two of these were formal evaluations with potential users of the simulation and others were less formal review and feedback sessions conducted in concert with our Agile development approach (*cf.* Schwaber and Beedle 2002).

3.1 Formal Evaluations

Formal evaluations of RACE were conducted in 2016 and 2017. In these evaluations, U.S. Air Force instructors were asked to use the RACE software to complete authoring (2016/17) or coaching (2017) tasks. Their performance was monitored to identify aspects of the process that could be improved. Further insights were gathered via participant surveys and debrief discussions. Indicative results are shown in Table 1. Generally, the participants felt strongly about the need for the type of independent practice offered by RACE, and they were relatively happy with the ease of the authoring process.

Table 1. Representative results from RACE evaluations

	2016	**2017**
Authoring Ease		
Coaching Value		

3.2 Informal Evaluation

Despite the overall favorable results of the evaluations, the development team was left with the impression that improvement was possible. One source of this impression was the debrief comments made by participants at the conclusion of the formal evaluations. A more powerful source was the less formal comments that we received at the end of each development sprint. In this section, we review some of those findings and the steps that we took to enhance the performance of the system.

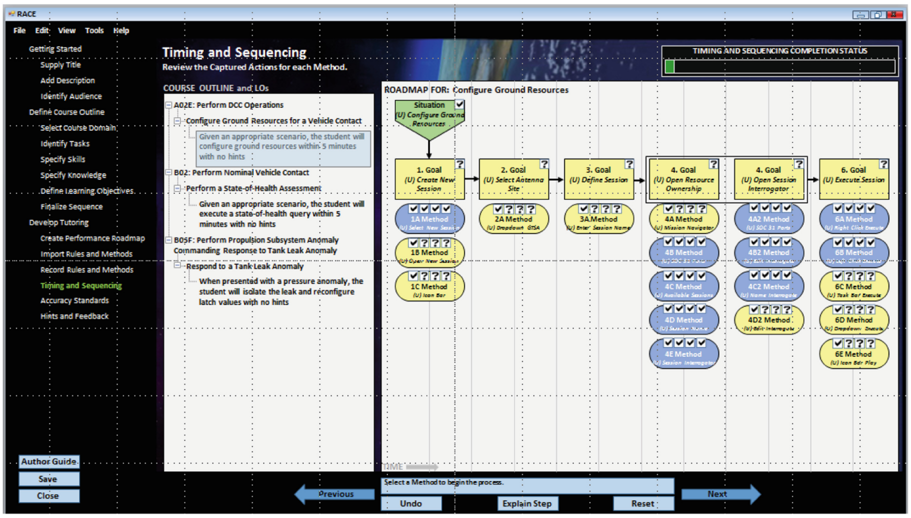
Authoring Approach was Too Restrictive. As the development team studied the feedback, certain trends emerged. First, it was clear that authors wanted to be able to define multiple correct procedures for many tasks, but that the current approach to doing so felt cumbersome. They wanted an easier way to provide students with the flexibility to complete tasks in a variety of ways. Second, because procedures were frequently repeated in support of various tasks, they wanted an ability to easily re-use the procedures previously demonstrated. Third, in some cases, task procedures had complex timing and sequential patterns. There was a concern that RACE wasn't sufficiently expressive to support user needs as we moved from a prototype to a fielded system.

To address these challenges the design team undertook an investigation regarding the ability to create a more graphical "front end" to the authoring process. Our goal was not to implement this approach, but to determine if we could create a design concept that felt more natural to users. Sonalysts developed a "Performance Roadmap" approach to authoring. With this approach, the author would select an LO and then use a visual workspace and icons to build a sequence of goals that would satisfy that learning objective. Each goal, in turn, would have one or more methods associated with it. A method would be a path for a response – a series of steps to accomplish a specific task or satisfy a goal. This approach would allow authors the flexibility to build many ways to allow students to perform a task, answering one of the usability concerns.

After building a timeline, an author would "populate" method blocks one of two ways. First, an author could import an existing method from an ever-growing RACE library. This would allow the re-use of "building blocks," answering the second of the usability concerns. Second, an author could demonstrate new methods using a simulation linked to the learning objective. Besides being saved to the current roadmap, newly recorded methods would be saved to the RACE library and become available for import into future roadmaps. After populating method blocks, an author would individually adjust timing and accuracy standards for each using a visual interface. A representation of this approach is shown in Fig. 2.

Concern About Coaching Level and Associated Level of Effort. With our new, more graphical authoring approach in hand, the development team reviewed it with a focus group. The user population expressed continued concern about the level of effort associated with the authoring task and their ability to support that level while doing their "day job." To help reduce the level of effort, the audience suggested doing away with methods and focusing instead on outcomes (*e.g.,* "we only care if they get the job done, not how they do it."). In reaction to this feedback, the development team undertook parallel analysis tasks. One group investigated whether an outcome-based approach was practical. The second group investigated a mechanism for simplifying the authoring process while maintaining process-oriented methods.

The development team concluded that a purely "outcome-based" approach to supporting independent practice was possible, but probably not advisable. There were several reasons for this determination. First, although the SST environment included the level of detail necessary to define required outcomes, we could not be sure that was true


RACE
File Edit View Tools Help
Getting Started
Supply Title
Add Description
Identify Audience
Define Course Outline
Select Course Domain
Identify Tasks
Specify Skills
Specify Knowledge
Define Learning Objectives
Finalize Sequence
Develop Tutoring
Create Performance Roadmap
Import Rules and Methods
Record Rules and Methods
Timing and Sequencing
Accuracy Standards
Hints and Feedback
Author Guide
Save
Close
TIMING AND SEQUENCING COMPLETION STATUS
ROADMAP FOR: Configure Ground Resources
Situation (U) Configure Ground Resources
1. Goal (U) Create New Session
2. Goal (U) Select Antenna Site
3. Goal (U) Define Session
4. Goal (U) Open Resource Ownership
4. Goal (U) Open Session Interrogator
6. Goal (U) Execute Session
1A Method
1B Method
1C Method (U) Icon Bar
2A Method (U) Dropdown GTSA
3A Method (U) Enter Session Name
4A Method (U) Mission Navigator
4B Method
4C Method (U) Available Sessions
4D Method (U) Session Name
4E Method (U) Session Interrogator
4A2 Method (U) SOC 31 Ports
4B2 Method
4C2 Method (U) Name Interrogate
4D2 Method (U) Edit Interrogate
6A Method (U) Right Click Execute
6B Method
6C Method (U) Task Bar Execute
6D Method (U) Dropdown Execute
6E Method (U) Icon Bar Play
Previous
Next
Undo
Explain Step
Reset

Timing and Sequencing
Review the Captured Actions for each Method.

COURSE OUTLINE and LOs

A02E: Perform DCC Operations
Configure Ground Resources for a Vehicle Contact
Given an appropriate scenario, the student will configure ground resources within 5 minutes with no hints
B02: Perform Nominal Vehicle Contact
Perform a State-of-Health Assessment
Given an appropriate scenario, the student will execute a state-of-health query within 5 minutes with no hints
B05F: Perform Propulsion Subsystem Anomaly Commanding Response to Tank Leak Anomaly
Respond to a Tank Leak Anomaly
When presented with a pressure anomaly, the student will isolate the leak and reconfigure latch values with no hints

Select a Method to begin the process.

Fig. 2. Representation of graphical roadmap approach to authoring

for all simulation environments and adopting this approach might violate our goal of being simulation-agnostic. Second, the very richness of the SST data model that made outcome-based assessment possible might make developing outcome recognition rules complex. As a result, the reduction in the level of effort that authors were seeking might not be realized. Third, the outcome-based strategy would limit the breadth of coaching that was available to whether or not the outcome was achieved, not why. As a result, the system would have a much diminished ability help to the students to improve.

Along the other path (simplifying process-based authoring), the development team proceeded by making a few simplifying assumptions. The primary simplification was to replace the "task-skill-knowledge-LO" decomposition process that would support both skill- and knowledge-focused instruction and practice, with a scenario-goal-method decomposition focused on supporting skill-based practice. Within this framework, the authoring process would have two major tasks. First, the author would decompose each scenario in the practice curriculum into a sequential series of useable goals. Second, the author would demonstrate the methods associated with each goal. By limiting the demonstration process to one goal at a time, this approach avoided the combinatorial explosion among methods that could sometimes occur within the LO-centric approach.

To explore this design concept, the design team proceeded along two paths. First, we wanted to see if this approach could produce high-value coaching. To explore this, we created a practice curriculum comprising eight scenarios that spanned an introductory curriculum in space operations. An SME analyzed the procedures associated with each practice activity and decomposed them into a set of relatively small goals that had a high degree of re-use across the practice activities. This process produced approximately 56 small re-usable goals. In addition, the SME created a sequence chart that illustrated when each goal could be completed. Figure 3 illustrates the outcome for a relatively simple and a relatively complex practice activity.

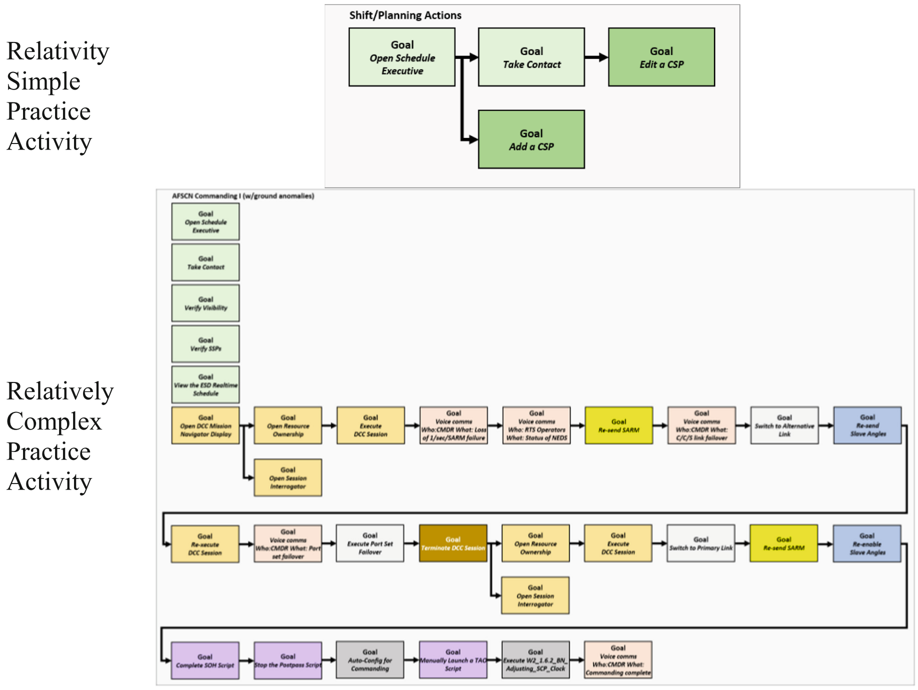

Relativity Simple Practice Activity
Shift/Planning Actions
Goal Open Schedule Executive
Goal Take Contact
Goal Edit a CSP
Goal Add a CSP
Relatively Complex Practice Activity
AFSCN Commanding I (w/ground anomalies)
Goal Open Schedule Executive
Goal Take Contact
Goal Verify Visibility
Goal Verify SSPs
Goal View the ESD Realtime Schedule
Goal Open DCC Mission Navigator Display
Goal Open Resource Ownership
Goal Execute DCC Session
Goal Voice comms Who:CMDR What: Loss of 1/sec/SARM failure
Goal Voice comms Who: RTS Operators What: Status of NEDS
Goal Re-send SARM
Goal Voice comms Who:CMDR What: C/C/S link failover
Goal Switch to Alternative Link
Goal Re-send Slave Angles
Goal Open Session Interrogator
Goal Re-execute DCC Session
Goal Voice comms Who:CMDR What: Port set failover
Goal Execute Port Set Failover
Goal Terminate DCC Session
Goal Open Resource Ownership
Goal Execute DCC Session
Goal Switch to Primary Link
Goal Re-send SARM
Goal Re-enable Slave Angles
Goal Open Session Interrogator
Goal Complete SOH Script
Goal Stop the Postpass Script
Goal Auto-Config for Commanding
Goal Manually Launch a TAC Script
Goal Execute W2_1.6.2_BN_ Adjusting_SCP_Clock
Goal Voice comms Who:CMDR What: Commanding complete

Fig. 3. Sample goal sequences

We then updated the RACE software to allow us to populate these goals with demonstrated methods and replicate these goal sequences within the coaching sequence. To approximate the outcome-oriented coaching that the instructors preferred, we disabled most of the evaluation standards (*e.g.,* we no longer assessed pacing or the parameters that students entered). We then asked our SME to review the resultant performance assessment and coaching. These "pilot tests" allowed us to make further refinements into how the RACE coaching engine responded to errors, how the system recognized that the students thought they were done, and how the system summarized student performance following the end of an activity.

Second, the development team replaced the graphical authoring approach discussed earlier with one that provided explicit support for defining the methods associated with each goal and sequencing those goals to support a given practice activity. The revised approach asked authors to demonstrate the methods for one goal at a time. In doing so, it avoided the sense of "overwhelming" instructors. Further, developing the roadmap was simply a process of importing completed goals. To provide a little more flexibility, we also allowed authors to disable methods if they felt that they were not appropriate in a given setting and to adjust the performance standards that would be enforced within a given practice activity. The result seems to strike a good balance between providing both expressive flexibility and simplicity.

4 Discussion

The RACE project explored the ability to develop authoring tools that would make it simpler to develop and maintain intelligent tutoring systems that can enable independent student practice. From a technology perspective, the development team felt that they were successful. The most daunting technical challenge was developing a generalized and usable approach to situation assessment. We made progress on this front, but ultimately other factors (specifically, the explicit definition of goal sequences to simplify other facets of authoring) obviated the need for a complete solution. This is an area ripe for further exploration in a cross-simulation/cross-domain environment.

A second challenge was the need to make the authoring as simple and intuitive as possible for our authors while maintaining the full power of an ITS. Our efforts here were helped by the inclusion of formal usability assessments within the effort. Perhaps more strikingly, our efforts were also enhanced by more periodic feedback afforded by the monthly development "sprints" used within these efforts. The Agile development approach allowed us to experiment with ideas and to get timely internal/external feedback on what "worked" while there was still time to benefit from the lessons learned. We plan to combine these two methods in most of our future research and development efforts.

Across the span of the prototyping effort, the development team feels that they have made significant progress in demonstrating the ability to field an authoring tool that is agnostic to both the hosting simulation environment and the tutoring engine and that produces high-quality coaching. We look forward to the opportunity to continue to test and refine this emerging technology.

References

Aleven, V., Mclaren, B.M., Sewall, J., Koedinger, K.R.: A new paradigm for intelligent tutoring systems: example-tracing tutors. Int. J. Artifi. Intell. Educ. **19**(2), 105–154 (2009)

Amant, R.S., Freed, A.R., Ritter, F.E.: Specifying ACT-R models of user interaction with a GOMS language. Cogn. Syst. Res. **6**(1), 71–88 (2005)

Cohen, M.A., Ritter, F.E., Haynes, S.R.: Herbal: a high-level language and development environment for developing cognitive models in Soar. In: Proceedings of the 14th Conference on Behavior Representation in Modeling and Simulation, pp. 133–140 (2005)

Corbett, A.T., Koedinger, K.R., Anderson, J.R.: Intelligent tutoring systems. In: Helander, M.G., Landauer, T.K., Prabhu, P.V. (eds.) Handbook of Human-Computer Interaction. Elsevier Science B.V., Amsterdam (1997)

Fletcher, J.D.: Intelligent training systems in the military. In: Andriole, S.J., Hopple, G.W. (eds.) Defense Applications of Artificial Intelligence: Progress and Prospects. Lexington Books, Lexington (1988)

Gott, S.P., Kane, R.S., Lesgold, A.: Tutoring for transfer of technical competence. Air Force Technical Report: AL/HR-TP-1995-0002. Armstrong Laboratory, Human Resources Directorate, Brooks AFB, TX (1995)

Katz, S., Lesgold, A.: The role of the tutor in computer-based collaborative learning situations. In: Lajoie, S.P., Derry, S.J. (eds.) Computers as Cognitive Tools. Erlbaum, Hillsdale (1993)

Koedinger, K.R., Aleven, V., Heffernan, N.: Toward a rapid development environment for Cognitive Tutors. In: Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies, Proceedings of AI-ED 2003. IOS Press (2003)

Lesgold, A., Lajoie, S., Bunzo, M., Eggan, G.: SHERLOCK: a coached practice environment for an electronics troubleshooting job. In: Larkin, J., Chabay, R. (eds.) Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches. Lawrence Erlbaum Associates, Hillsdale (1992)

McCarthy, J.E.: Military applications of adaptive training technology. In: Lytras, M.D., Gaševiće, D., Ordóñez de Pablos, P., Huang, W. (eds.) Technology Enhanced Learning: Best Practices. IGI Publishing, Hershey (2008)

Nullmeyer, R., Bennett, W.: Leading-Edge Training Research is Catalyst for Training Transformation in Space Operations. Fight's On **7**(1), 1 (2008)

Ritter, S., Koedinger, K.R.: Towards lightweight tutoring agents. In: Proceedings of AI-ED 1995 - World Conference on Artificial Intelligence in Education, Washington, DC, pp. 91–98, August 1995

Schwaber, K., Beedle, M.: Agile Software Development with Scrum, vol. 1. Prentice Hall, Upper Saddle River (2002)