Design Principles for Pedagogical Agent Authoring Tools

James Lester, Bradford Mott, Jonathan Rowe and Robert Taylor
Center for Educational Informatics
North Carolina State University

INTRODUCTION

Pedagogical agents hold great promise for enhancing the learning experience of students within intelligent tutoring systems (ITSs). There is mounting evidence that ITSs lead to improved student learning (Beal, Walles, Arroyo, & Woolf, 2007; Schroeder, Adesope, & Gilbert, 2013) and in some cases have been found to be nearly as effective as one-on-one human tutoring (VanLehn, 2011). The timely and customized advice of ITSs may be further enhanced by the addition of pedagogical agents embodied as virtual characters that have the ability to motivate students while simultaneously providing complementary feedback through deictic gestures, motions, and utterances (Lester, Voerman, Towns, & Callaway, 1999; Rus, D'Mello, Hu, & Graesser, 2013). Advancing the case for employing pedagogical agents in tutoring systems is the increase in availability of game engines and graphics hardware capable of rendering lifelike virtual characters with significantly reduced development effort (Petridis et al., 2012).

Despite the potential for increased student engagement and the reduced cost of creating lifelike virtual characters, pedagogical agents have not yet achieved widespread adoption in computer-based learning environments. A formidable and well-known barrier to building and widely deploying a pedagogical agent is the complexity and expense associated with instilling the pedagogical agent with domain-specific knowledge and tutoring strategies (Murray, 2003; Woolf, 2009). Furthermore, an additional complication in creating an effective pedagogical agent is that the agent must present believable, lifelike behaviors such that students feel they are observing and interacting with "a sentient being with its own beliefs, desires, and personality" (Lester & Stone, 1997). Thus, a limiting factor in the widespread deployment of pedagogical agents is the significant effort and pedagogical agent expertise required to codify knowledge and behaviors from subject matter experts into the ITS.

An approach to solving this problem is improving the efficiency of codifying expert knowledge by creating pedagogical agent authoring tools that are tailored for subject matter experts rather than researchers. However, creating an effective authoring tool for subject matter experts poses two principal challenges. First, it must facilitate the creation of curricular content for the learning environment by subject matter experts who are not pedagogical agent experts and are often not software engineers. Second, it must support the creation or modification of pedagogical agent behaviors without exposing the complexity of the pedagogical agent itself to the subject matter expert. In practice, a majority of the design and programming effort expended on pedagogical agents is developing the agent and the learning environment itself. This often results in the authoring tool being treated as an afterthought, leaving little time or resources to design and develop authoring tools that are suitable for subject matter experts. Based on our experience developing a pedagogical agent authoring tool for educators, this chapter identifies promising authoring tool principles and features that could improve the authoring efficiency of subject matter experts. To conclude, we reason that the Generalized Intelligent Framework for Tutoring (GIFT) (Sottilare, Brawner, Goldberg, & Holden, 2012) could be used to provide a high-quality implementation of these authoring tool design principles and, therefore, act as a force multiplier for creating new pedagogical agent-based tutoring systems that utilize GIFT.

RELATED RESEARCH

Creating authoring tools for building ITSs is receiving ever-increasing attention from the research community. With a goal of making ITS creation and authoring accessible to subject matter experts who are not computer scientists, progress is being made in researching approaches to create authoring tools (Susarla, Adcock, Van Eck, Moreno, & Graesser, 2003; Jordan, Hall, Ringenberg, Cue, & Rose, 2007) and automate aspects of pedagogical agents such as dialogue (André et al., 2000; Si, Marsella, & Pynadath, 2005; Piwek, Hernault, Prendinger, & Ishizuka, 2007) or nonverbal behaviors (Lhommet & Marsella, 2013).

Authoring tools for conversation-based learning environments have focused on assisting non-technical users in the creation of pedagogical agent dialogues. AutoTutor provides multi-agent conversational interactions to tutor students using the discourse patterns of a human tutor. AutoTutor has been used across multiple domains including computer literacy and physics (Graesser, Chipman, Haynes, & Olney, 2005). To facilitate the application of AutoTutor to other domains, authoring tools have been developed to aid subject matter experts in creating dialogue-based tutors, such as the AutoTutor Script Authoring Tool (Susarla, Adcock, Van Eck, Moreno, & Graesser, 2003) and AutoLearn (Preuss, Garc, & Boullosa, 2010). Another example of an authoring tool for agent dialogue is TuTalk, which was created to support the rapid development of agent-based dialogue systems by non-programmers (Jordan, Hall, Ringenberg, Cue, & Rose, 2007). This tool facilitates the authoring of domain knowledge and resources required by the dialogue agent in the form of AI planning techniques that address high-level goals of the dialogue system. Similarly, an authoring tool has been created for the Tactical Language and Culture Training System (TLCTS) that allows subject matter experts to create pedagogical dialogue for a foreign language learning training system at reduced cost and time (Meron, Valente, & Johnson, 2007).

Another approach to improving pedagogical agent authoring is to remove the need for authoring altogether through the use of automation. In particular, automating the creation of pedagogical agents' lifelike nonverbal behaviors eliminates a potentially significant amount of authoring effort. Cerebella is a system that monitors an agent's utterances (in both text and audio formats) and automatically generates lifelike nonverbal behaviors such as averting gaze, raising an eye brow, or slumping shoulders (Lhommet & Marsella, 2013). The automatically generated nonverbal behaviors inferred from the communicative intent and underlying mental state of the agent can be used as an additional channel of communication between the pedagogical agent and the student, increasing the agent's believability as well as students' engagement. The THESPIAN system reduces the effort to author pedagogical agents by facilitating the creation of interactive pedagogical dramas (Si, Marsella, & Pynadath, 2005). In THESPIAN, the learner and the pedagogical agents interact with each other as characters within a story. THESPIAN accepts as input a set of scripts that it uses to automatically generate and adjust agents' goals to guide their behavior. Another example of automating pedagogical agent authoring tasks is to convey domain knowledge to the student through observations of simulated conversations and interactions between agents. The agent dialogue, character selection, and content rendering tasks would be automatically performed by the presentation system as described by André et al (2000). In this approach, information is communicated by decomposing knowledge into atomic information units that are then conveyed to the student through verbal and nonverbal interactions between two or more agents.

Authoring of pedagogical agents can be accelerated by leveraging knowledge that has already been recorded in other forms such as Wikipedia pages, PowerPoint presentations, dialogue scripts, or PDFs. The Tools for Rapid Automated Development of Expert Models (TRADEM) project parses existing domain content and automatically generates dialogue, questions, and a script that represents the order of instruction based on the ordering of the original content (Robson, Ray, & Cai, 2013). This system can be used to create a minimal dialogue-based tutoring system where a pedagogical agent can ask questions and evaluate student answers related to the original content without requiring a subject matter expert to explicitly author the knowledge or assessments in the ITS (Brawner & Graesser, 2014). Text2Dialogue is another system that can utilize existing knowledge represented as text files to produce dialogue that is acted out by 3D virtual characters (Piwek, Hernault, Prendinger, & Ishizuka, 2007). A significant difference between this approach and the previously described presentation system developed by André et al (2000) is that Text2Dialogue can accept textual information as input without presentation goals being defined by a subject matter expert, which means that dialogue may be generated from existing text files without requiring annotation by a subject matter expert.

Even though the aforementioned research into implementing, augmenting, and eliminating the need for pedagogical agent authoring tools holds great promise, there is still an immediate need for effective and efficient tools that enable subject matter experts to codify knowledge and tutoring strategies as pedagogical agents without requiring the subject matter experts to posses or acquire programming or intelligent tutoring expertise.

DISCUSSION

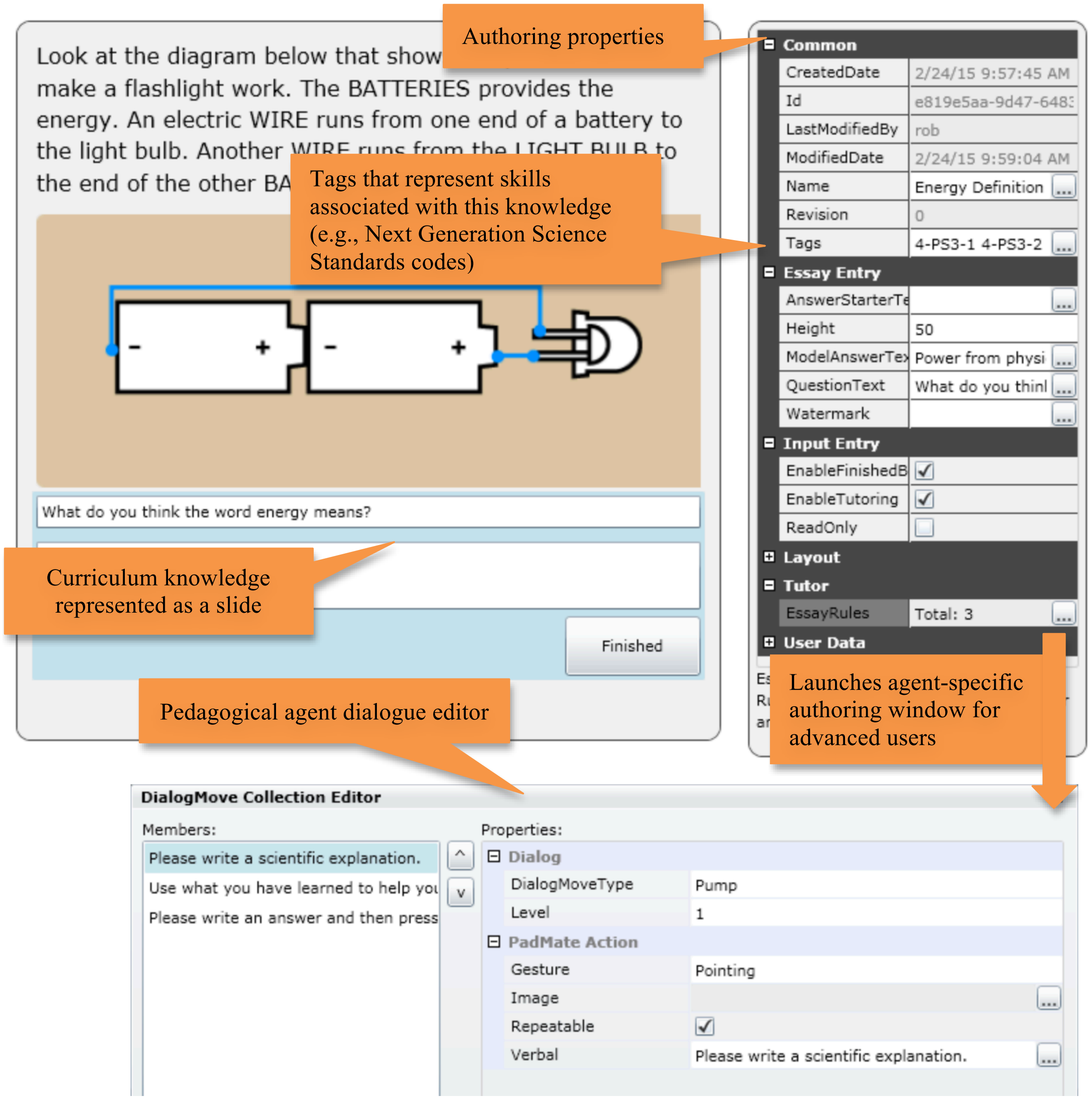
To address the immediate need for effective and efficient authoring tools, we present seven design principles that are grounded in software engineering practice and have the potential to significantly improve pedagogical agent authoring tools intended for subject matter experts. We illustrate our discussion with the COMPOSER authoring tool, which was developed for non-technical subject matter experts to author pedagogical agents. We describe our lessons-learned utilizing COMPOSER to create a pedagogical agent for a widely deployed ITS for upper elementary science education.

Design Principles for Pedagogical Agent Authoring Tools

To make pedagogical agent-based learning environments more widely available, authoring tools must be designed and implemented that empower subject matter experts to quickly and efficiently populate the domain knowledge and tutoring strategies used by the pedagogical agent. To this end, creating usable and efficient authoring tools can be framed as a software engineering problem that may be addressed by general software design principles. The principles we advocate are well established in software engineering. Our contribution is discussing how to operationalize these principles in the context of authoring pedagogical agents. Since the design and implementation of authoring tools directly impacts the design and implementation of intelligent pedagogical agents (and vice versa), we recommend that the following pedagogical agent authoring tool design principles be considered at the beginning of a project, and leveraged in concert with the development of the learning environment, rather than leaving the tool development for the end of the project, where the tool will be constrained by an existing pedagogical agent implementation. In this section, we enumerate software design principles and features that should be considered for inclusion in a subject matter expert-centered pedagogical agent authoring tool.

Adopt a Familiar User Interface Paradigm. From a usability standpoint, the most important feature of an authoring tool is its user interface (UI). Ideally, a pedagogical agent authoring tool should present a UI that is familiar and intuitive for the type of subject matter expert who is intended to use it. Instead of requiring the subject matter expert to conform to unfamiliar ITS naming conventions and authoring workflow, the authoring tool should be modeled after software that the subject matter expert is already comfortable using. For example, if the intended user of the tool is a K-12 teacher, this type of user is likely very comfortable using Microsoft PowerPoint to create presentations to be shown in the classroom. Likewise, if the type of subject matter expert is a computer scientist, this user will be comfortable writing code and using an integrated development environment (IDE), such as Eclipse. Of course, existing UIs and usage paradigms can (and should) be improved upon; however, instead of starting from scratch when designing an authoring tool, modeling upon an existing tool leverages decades of real-world usability and efficiency improvements.

Modeling a pedagogical agent authoring tool's UI after an existing authoring tool, such as Microsoft PowerPoint, does not imply that the tutoring knowledge constructs must be as simple as the content in a typical PowerPoint presentation. This would indeed be challenging since tutoring systems are likely to require authoring pedagogical strategies or annotating answer correctness, which are features that are not afforded by the PowerPoint UI. Instead, this principle implies that the authoring tool should model the existing tool by using similar naming conventions, presenting similar software features, and mimicking its workflow. For example, a pedagogy-oriented authoring tool might represent blocks of curriculum knowledge as "slides" in a PowerPoint-like authoring tool (Figure 1). Likewise, a slide might provide static text or multimedia that is used to convey information to the student, as well as embedded assessments that are used to gauge student competencies. Slides could be associated with production rules


Authoring properties

Look at the diagram below that show[s] make a flashlight work. The BATTERIES provides the energy. An electric WIRE runs from one end of a battery to the light bulb. Another WIRE runs from the LIGHT BULB to the end of the other BA[TTERY]

Tags that represent skills associated with this knowledge (e.g., Next Generation Science Standards codes)

What do you think the word energy means?

Curriculum knowledge represented as a slide

Pedagogical agent dialogue editor

Launches agent-specific authoring window for advanced users

Common	
CreatedDate	2/24/15 9:57:45 AM
Id	e819e5aa-9d47-6483
LastModifiedBy	rob
ModifiedDate	2/24/15 9:59:04 AM
Name	Energy Definition
Revision	0
Tags	4-PS3-1 4-PS3-2
Essay Entry	
AnswerStarterTe	
Height	50
ModelAnswerTex	Power from physi
QuestionText	What do you thinl
Watermark	
Input Entry	
EnableFinishedB	✓
EnableTutoring	✓
ReadOnly	
Layout	
Tutor	
EssayRules	Total: 3
User Data	

DialogMove Collection Editor

Members:

- Please write a scientific explanation.
- Use what you have learned to help you
- Please write an answer and then press

Properties:

Dialog	
DialogMoveType	Pump
Level	1
PadMate Action	
Gesture	Pointing
Image	
Repeatable	✓
Verbal	Please write a scientific explanation.

Figure 1. Slide-based authoring paradigm illustrated by the COMPOSER authoring tool user interface

for teaching the specific concepts and skills represented by the slide, while tags enable the pedagogical agent to associate students' performance with an overarching set of knowledge components. Without the subject matter expert explicitly authoring it, the pedagogical agent could use this metadata and the student model to determine the next slide to display to the student.

Include Standard Editing Features. Modeling a pedagogical agent authoring tool after a mature software package, such as Microsoft PowerPoint, suggests the implementation of several software features which are expected and relied upon by typical software users; however, these features are often nontrivial to implement and have profound effects on how data is represented, stored, and manipulated within the authoring tool, which is likely to affect how the data is represented in the ITS itself. For example, copy, cut, and paste features are expected by users to be available on any data type that can be authored in a tool. This feature may require deep or shallow copies of data models used to represent curriculum and pedagogical data while maintaining relationships between the data. Similarly, the undo and redo features enable users to experiment and quickly repair authoring mistakes. Undo and redo can drastically impact the design and implementation of the authoring tool itself and, therefore, should not be left as a feature to be added at the end of project when there is limited time to refactor data models or add revision tracking to the content being authored.

Support Author Collaboration. A pedagogical agent authoring tool should implement features that allow multiple subject matter experts to collaborate while authoring domain knowledge and pedagogical strategies. Collaboration has the potential to increase both the quality and quantity of content available to the ITS. Users have come to expect and rely upon collaboration features in other contexts. For example, at one extreme, multiple authors can use web browsers to simultaneously edit a single Google document, presentation, or spreadsheet. The authors can view each other's modifications and chat with one another while editing. Likewise, many content authoring tools enable change tracking to record which author made a change and when, or allow an author to comment on a piece of content without changing it in the form of a note. Implementing collaboration in an authoring tool will have significant impacts on the design of data models, the architecture of the application, and user authorization in regards to who is allowed to access which data. For example, storing domain knowledge and pedagogical strategies in a cloud-based server and implementing a web browser-based authoring tool would simplify implementation of collaboration features. Of course, this decision would need to be considered early in the design of the pedagogical agent and the authoring tool since it would impact the architecture and implementation of the entire system.

Facilitate Rapid Iteration and Testing. To facilitate refining the domain knowledge and pedagogical agent behaviors, the authoring tool should support a "rapid iteration" mode where small changes made in the authoring tool can be quickly seen and interacted with in the context of the ITS. In this mode, the subject matter expert can ideally interact with the pedagogical agent while editing content in real-time or with only a minor delay. This feature allows the subject matter expert to quickly confirm that content is presented in a visually appealing manner in the learning environment and that the pedagogical agent behaves in a believable manner while the subject matter expert is modifying properties or settings that influence the pedagogical agent's behavior. This feature could be implemented as a real-time connection to the ITS running as a separate application or the ITS could be embedded in the authoring tool to provide a WYSIWYG experience. In either situation, the data models would be required to support incremental dynamic updates and the ITS itself would have to respond to commands from the authoring tool such as navigating to specific domain content or modify the current state of the pedagogical agent depending on the types of edits the subject matter expert is making.

Accommodate Novice and Expert Authors. The pedagogical agent authoring tool should support editing methods that are specifically tailored to novice and expert users rather than presenting a one-sizefits-all UI. For example, a novice user is likely to be overwhelmed and discouraged by an authoring tool that exposes too many ITS-specific properties or settings. Conversely, an expert will be less efficient and will be frustrated by a UI that repeatedly walks through a series of basic steps. Therefore, for less frequently used authoring activities, or when authoring complex knowledge representations or pedagogical agent-specific behavior, the authoring tool should present a step-by-step wizard interface for novice users and a more direct authoring UI for expert users. For example, when authoring rules to evaluate the answer to an essay question, a wizard UI might ask the subject matter expert a series of questions that are used to generate a set of rules for evaluating the answer. On the other hand, an expert user would have the option of bypassing the wizard and authoring the rules directly. Interestingly, this design principle could be realized by embedding a pedagogical agent in the authoring tool itself to assist the subject matter expert in authoring content.

Automate Complex and Tedious Tasks. Some aspects of authoring domain knowledge or pedagogical agent behavior may be too complicated, labor intensive, or tedious for a subject matter expert to accomplish manually using a pedagogical agent authoring tool. In these situations, the authoring tool should provide automated mechanisms for generating curriculum content, pedagogical strategies, and pedagogical agent behaviors. This is where the automatic agent behavior generation techniques, as illustrated by Cerebella (Lhommet & Marsella, 2013), and automatic dialogue generation methods, as leveraged by THESPIAN (Si et al., 2005), can be utilized within the pedagogical agent authoring tool to reduce the authoring load for a subject matter expert.

Another approach to simplify the authoring of knowledge and pedagogical strategies is to assist the subject matter expert though the use of data mining techniques. Instead of authoring a pedagogical agent with strategies for every conceivable situation, authoring effort could be placed on the most common misconceptions or areas where students are showing weakness. In an educational data mining study by Merceron and Yacef, student data from a web-based learning environment was mined to inform teachers about students who were at risk (Merceron & Yacef, 2005). Students were grouped into learner cohorts using clustering techniques to identify students who were having difficulties. In a similar way, an ITS could initially be deployed with curriculum content but a relatively primitive pedagogical agent. After collecting student answers, the data could be mined to identify common misconceptions or domain knowledge that may require additional scaffolding by the pedagogical agent. The authoring tool would flag sections of the domain knowledge or identify broader concepts that the subject matter expert should focus on improving. This would naturally lead to an iterative authoring process where the pedagogical agent continues to evolve by focusing effort on the issues most relevant to students who are using the ITS. Using this type of authoring assistance feature has the potential to dramatically reduce the amount of authoring effort, because the subject matter expert is not required to exhaustively predict and annotate all possible correct and incorrect answers. On the other hand, the initial iterations of the pedagogical agent are unlikely to be particularly effective since they will have limited ability to provide remediation to students who are having difficulty.

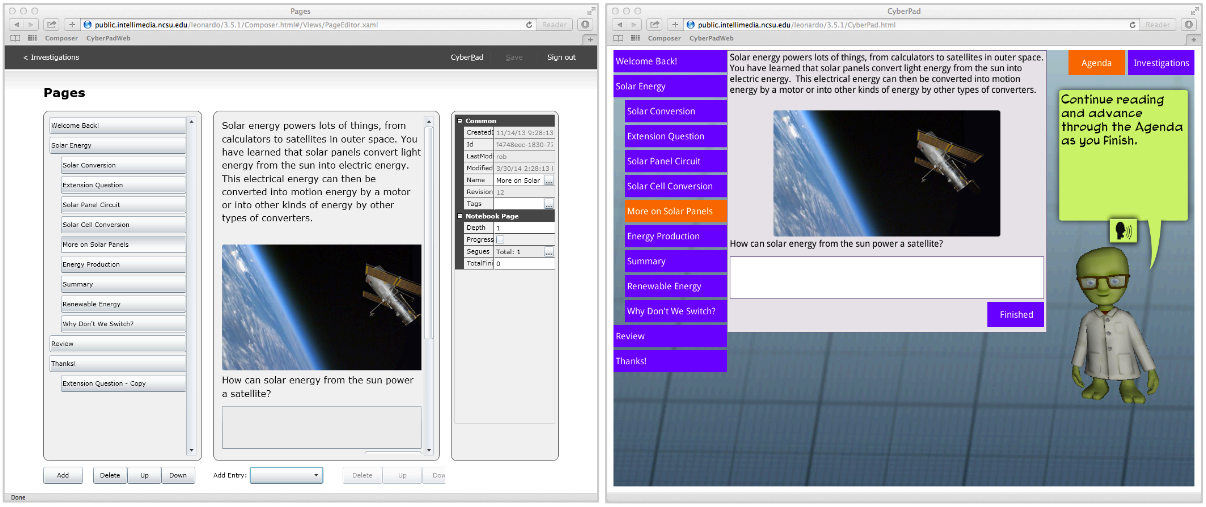
Avoid the Blank Page. Pedagogical agent authoring tools should assist the subject matter expert in getting started. Starting from a blank page using an unfamiliar tool can be a daunting task for any author of any skill level. This is particularly the case for someone who is authoring content for software as complex as a pedagogical agent-based ITSs. Therefore, authoring tools should provide templates and sample systems that can be used as starting points for authoring domain knowledge and pedagogical agent behaviors and dialogue. In addition, allowing subject matter experts to easily share their work with others has the potential to create a community that can evolve pedagogical agents by starting with another author's agent and building upon it rather than starting from scratch.

Importing existing knowledge that is already authored in the form of Microsoft Word documents, web pages, PowerPoint presentations, databases, or text files is a powerful feature for authoring tools to assist subject matter experts in quickly moving past the blank page. Taking it several steps further, automated systems such as TRADEM (Robson et al., 2013) and Text2Dialogue (Piwek et al., 2007) import existing knowledge and then automatically author agent dialogue, further reducing (or possibly eliminating) the pedagogical agent authoring load on the subject matter expert.

Lessons Learned from the LEONARDO Digital Science Notebook

For the past four years our laboratory has been developing a digital science notebook for upper elementary science education, the LEONARDO CyberPad, which runs on the Apple iPad and within web browsers on Windows and Mac OS X computing platforms. LEONARDO integrates a pedagogical agent into a digital science notebook that enables students to graphically model science phenomena. With a focus on the physical and earth sciences, the LEONARDO PadMate, a 3D embodied pedagogical agent, supports students' learning with real-time problem-solving advice. LEONARDO's curriculum is based on the Full Option Science System (Mangrubang, 2004). Throughout the inquiry process, students using the LEONARDO CyberPad are invited to answer multiple-choice questions, write answers to constructed response questions, and create symbolic sketches of different types, including electrical circuits. To date, LEONARDO has been implemented in over 70 elementary school classrooms across the United States.

LEONARDO consists of three major components: the CyberPad digital science notebook, the COMPOSER authoring tool, and a cloud-based server. Fourth and fifth grade elementary students learn about magnetism, circuits, and electricity using the CyberPad software (Figure 2, right). Subject matter experts use the COMPOSER (Figure 2, left) authoring tool to create curriculum content displayed in the digital science notebook, as well as rules, dialogue, and gestures that drive the pedagogical agent, which is embodied as a green alien within the CyberPad UI. The cloud-based server is used to store all curriculum knowledge, tutoring rules, and student data. During the design and development of the COMPOSER authoring tool, many of the principles for creating subject matter expert-centered authoring tools were identified as necessary features or enhancements that would improve the productivity of subject matter experts.



Solar energy powers lots of things, from calculators to satellites in outer space. You have learned that solar panels convert light energy from the sun into electric energy. This electrical energy can then be converted into motion energy by a motor or into other kinds of energy by other types of converters.

How can solar energy from the sun power a satellite?

Common	
CreatedD	11/14/13 9:28:13
Id	f4748eec-1830-77
LastMod	rob
Modified	3/30/14 2:28:13 I
Name	More on Solar
Revision	12
Tags	
Notebook Page	
Depth	1
Progress	☐
Segues	Total: 1
TotalFini	0

Solar energy powers lots of things, from calculators to satellites in outer space. You have learned that solar panels convert light energy from the sun into electric energy. This electrical energy can then be converted into motion energy by a motor or into other kinds of energy by other types of converters.

How can solar energy from the sun power a satellite?

Continue reading and advance through the Agenda as you Finish.

Figure 2. The COMPOSER tool (left) and CyberPad (right) in rapid iteration editing mode

The LEONARDO project did not originally include the COMPOSER authoring tool in its work plan. The first year of the project was spent designing and implementing a prototype of the CyberPad application to field test with fourth and fifth grade students to assess the practicality and ergonomics of using iPads in elementary school classrooms. During the first year, subject matter experts, who were science education faculty and graduate students, used Microsoft Word to author all of the curriculum content and pedagogical agent dialogue. The development team, who were computer science research staff and graduate students, manually copied the text from the Microsoft Word document into multiple XML documents. The XML documents were then embedded in the CyberPad iPad application as fixed resources that were then installed on iPads. The agent dialogue and rules were coded directly into the CyberPad's source code. Needless to say, this approach to content authoring was highly inefficient. It was labor intensive and error prone due to the need to repeatedly copy data by hand. In addition, pedagogical agent rules, dialogue, and gestures were tightly coupled with the contents of the XML documents making the entire system highly susceptible to syntax and typographical errors.

This initial approach to pedagogical agent authoring for the LEONARDO project had several significant drawbacks: First, the subject matter experts did not have a means to visualize what the curriculum content and pedagogical agent dialogue would look like when it was displayed in the CyberPad UI as they were authoring content in Microsoft Word. Second, it was extremely slow to make small changes to the content since it required a development team member to be available to a) make the change in XML, b) rebuild the application, and c) redeploy the CyberPad application to the iPads. Third, this dependency resulted in frustration for the subject matter experts and development team members. As a result, the curriculum content lacked polish, which is typically achieved by making many small changes after the original content is created. Since making small changes was highly inefficient, these changes were not made due to lack of resources and time. Using this approach to authoring content, one hour of instruction required more than the estimated 300 hours of development time often cited for ITS authoring (Tom Murray, 2003).

Based on this initial authoring experience and future plans to more than triple the amount of curricular content and pedagogical agent dialogue, it became imperative to design and implement the COMPOSER authoring tool in the second year of the project. We started requirements gathering by identifying the types of subject matter experts who would use the tool in the future: elementary school teachers, education graduate students, and education faculty. We then proceeded to design COMPOSER's UI by reviewing authoring tools from other areas that our subject matter experts were comfortable using. This included applications such as Microsoft PowerPoint, Google documents, and Edmodo. In the new system, curriculum content, agent dialogue, and rules would be stored in a cloud-based server where it could be directly accessed by both the COMPOSER tool and the CyberPad application. This approach formed the basis for the authoring tool principles and features proposed in this chapter.

The COMPOSER authoring tool improved the authoring workflow for the LEONARDO project in years two and three by decoupling content authors from the development team. Subject matter experts were empowered to refine curriculum content and pedagogical agent behavior independently of the development team. In addition, a familiar workflow and editing feature set further improved the efficiency of subject matter experts. However, since authoring wasn't considered in the initial design, these improvements did come at a development cost of refactoring data models, logic, and storage to make it possible to edit and track small discrete parts of the curriculum.

RECOMMENDATIONS AND FUTURE RESEARCH

Widespread development and deployment of pedagogical agents in intelligent tutoring systems depends on efficient transfer of domain knowledge and pedagogical agent dialogue, strategies, and behaviors from subject matter experts to the tutoring system. Authoring tools hold great promise to facilitate knowledge engineering. However, it should be emphasized that authoring tools should be tailored to the subject matter expert using features and workflows that have been proven effective by authoring software from non-ITS domains.

In future work, it will be important to investigate the addition of automation features to assist in the authoring of pedagogical behaviors and tutoring strategies. In the near term, leveraging educational data mining techniques to discover prevalent student behaviors, as well as misconceptions, from ITS datasets could further enhance ITS authoring tools and identify parts of curricula that require additional scaffolding. Future work should strive to immediately incorporate decades of software engineering knowledge in the design and implementation of novice and expert UIs to simplify authoring complex knowledge and underlying ITS mechanisms.

The design principles for pedagogical agent authoring tools presented in this chapter are not specific to any given tutoring system. Since these authoring tool principles are broadly applicable across ITSs, GIFT affords a unique opportunity to act as a authoring tool platform where many of these authoring tool design principles could be implemented once and used by many tutoring systems. This would allow a single high-quality authoring tool implementation to be established that could then be shared across multiple tutoring systems, thereby reducing redundant authoring tool design and development effort across multiple projects while simultaneously raising the quality of the authoring tools based on GIFT. It follows that this approach has the potential to produce higher-quality pedagogical agent-based learning environments more quickly and at reduced cost.

REFERENCE

Beal, C. R., Walles, R., Arroyo, I., & Woolf, B. P. (2007). On-line tutoring for math achievement testing: A controlled evaluation. *Journal of Interactive Online Learning*, *6*(1), 43–55.

Brawner, K., & Graesser, A. (2014). Natural Language, Discourse, and Conversational Dialogues within Intelligent Tutoring Systems: A Review. In R. Sottilare, A. Graesser, X. Hu, & B. Goldberg (Eds.), *Design Recommendations for Intelligent Tutoring Systems* (pp. 189–204).

Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005). AutoTutor: An Intelligent Tutoring System With Mixed-Initiative Dialogue. *IEEE Transactions on Education*, *48*(4), 612–618.

Jordan, P. W., Hall, B., Ringenberg, M., Cue, Y., & Rose, C. (2007). Tools for Authoring a Dialogue Agent that Participates in Learning Studies. In R. Luckin, K. R. Koedinger, & J. Greer (Eds.), *Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work* (pp. 43–50). IOS Press.

Lester, J. C., & Stone, B. A. (1997). Increasing believability in animated pedagogical agents. In *AGENTS '97 Proceedings of the First International Conference on Autonomous Agents* (pp. 16–21).

Lester, J. C., Voerman, J. L., Towns, S. G., & Callaway, C. B. (1999). Deictic Believability: Coordinated Gesture, Locomotion, and Speech in Lifelike Pedagogical Agents. *Applied Artificial Intelligence*, *13*(4-5), 383–414.

Lhommet, M., & Marsella, S. C. (2013). Gesture with meaning. In *Intelligent Virtual Agents* (pp. 303–312). Springer Berlin Heidelberg.

Mangrubang, F. R. (2004). Preparing elementary education majors to teach science using an inquiry-based approach: The Full Option Science System. *American Annals of the Deaf*, *149*(3), 290–303.

Merceron, A., & Yacef, K. (2005). Educational Data Mining: a Case Study. In *AIED* (pp. 467–474).

Meron, J., Valente, A., & Johnson, W. L. (2007). Improving the authoring of foreign language interactive lessons in the tactical language training system. In *Speech and Language Technology in Education (SLaTE2007)* (pp. 33–36).

Murray, T. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environments* (pp. 493–546). Springer Netherlands.

Petridis, P., Dunwell, I., Panzoli, D., Arnab, S., Protopsaltis, A., Hendrix, M., & de Freitas, S. (2012). Game Engines Selection Framework for High-Fidelity Serious Applications. *International Journal of Interactive Worlds*, *2012*, 1–19.

Piwek, P., Hernault, H., Prendinger, H., & Ishizuka, M. (2007). T2D: Generating Dialogues Between Virtual Agents Automatically from Text. In *Intelligent Virtual Agents* (pp. 161–174). Springer Berlin Heidelberg.

Preuss, S., Garc, D., & Boullosa, J. (2010). AutoLearn's Authoring Tool: A Piece of Cake for Teachers. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 19–27). Association for Computational Linguistics.

Robson, R., Ray, F., & Cai, Z. (2013). *Transforming Content into Dialogue-based Intelligent Tutors*. Paper presented at The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.

Rus, V., D'Mello, S. K., Hu, X., & Graesser, A. C. (2013). Recent advances in intelligent tutoring systems with conversational dialogue. *AI Magazine*, *34*(3), 42–54.

Schroeder, N. L., Adesope, O. O., & Gilbert, R. B. (2013). How Effective are Pedagogical Agents for Learning? A Meta-Analytic Review. *Journal of Educational Computing Research*, *49*(1), 1–39.

Si, M., Marsella, S. C., & Pynadath, D. V. (2005). THESPIAN: An Architecture for Interactive Pedagogical Drama. In *AIED* (pp. 595–602).

Sottilare, R. A., Brawner, K. W., Goldberg, B. S., & Holden, H. K. (2012). A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems (CBTS). In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference*.

Susarla, S., Adcock, A., Van Eck, R., Moreno, K., & Graesser, A. C. (2003). Development and evaluation of a lesson authoring tool for AutoTutor. In *AIED2003 supplemental proceedings* (pp. 378–387).

VanLehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, *46*(4), 197–221.

Woolf, B. P. (2009). *Building Intelligent Interactive Tutors: Student-centered strategies for revolutionizing e-learning*. San Francisco, CA: Morgan Kaufmann.