Towards ITS Authoring Tools for Domain Experts

Robert Taylor, Andy Smith, Samuel Leeman-Munk, Bradford Mott, and James Lester

North Carolina State University, Raleigh, North Carolina, USA
{rgtaylor, pmsmith4, spleeman, bwmott, lester}@ncsu.edu

Abstract. The scarcity of efficient and user-friendly authoring tools has long been acknowledged as a limiting factor in the widespread development and deployment of intelligent tutoring systems (ITSs). Creating an effective authoring tool for domain experts poses two significant challenges: it must facilitate the creation of curricular content by domain experts who are typically neither ITS experts nor software engineers, and it must support the creation or modification of ITS-specific pedagogical strategies without exposing the complexity of the ITS itself to the domain expert. This paper presents a set of authoring tool design principles such as leveraging existing UI workflows, collaboration, and automation to improve the effectiveness of domain experts. Specifically, this paper examines the design of ITS authoring tools through the lens of software engineering.

Keywords: Authoring tools, Intelligent tutoring systems, Software engineering.

1 Introduction

Intelligent tutoring systems (ITSs) hold great promise for enhancing the learning experience of students at all levels inside and outside the classroom [1]. ITSs have been used in various forms in laboratories, classrooms, and workplaces for more than forty years [2]. However, ITSs have not achieved widespread adoption despite evidence that they lead to improved student learning [3, 4] and in some cases have been found to be nearly as effective as one-on-one human tutoring [5]. The scarcity of ITSs in the classroom is even more remarkable given the ever increasing availability of network connectivity and computational power available to implement and deploy sophisticated tutoring systems [1].

A formidable and well-known barrier to building and widely deploying ITSs is the complexity and expense associated with developing an ITS and populating it with domain-specific content and tutoring strategies [6]. It has been estimated that up to 300 hours of development time are required to create one hour of instruction [1, 6]. To alleviate the expense and time required to build an ITS suitable for deployment outside the lab, new ITS techniques have been developed such as example-tracing tutors [7] and constraint-based tutors [8]. Even with these notable advances and the creation of a variety of ITS authoring tools [9], significant effort is still required to codify knowledge from domain experts (or subject matter experts) into the ITS.

A potential solution to this problem is to create ITS authoring tools that are tailored for the domain experts who will use them. However, creating an effective authoring tool for domain experts poses two significant challenges. First, it must facilitate the creation of curricular content for the ITS by domain experts who are not ITS experts and are often not software engineers. Second, it must support the creation or modification of ITS-specific pedagogical strategies without exposing the complexity of the ITS itself to the domain expert. In practice, a majority of the design and programming effort expended on an ITS is often spent on developing the ITS itself, which results in the authoring tool being treated as an afterthought, leaving little time and resources to design and develop a tool that is suitable for domain experts. Based on our experience of developing an ITS authoring tool for educators, this paper identifies promising authoring tool principles and features that could improve the authoring efficiency of domain experts.

2 Design Principles for ITS Authoring Tools

To make intelligent tutoring systems more widely available, ITS authoring tools must be designed and implemented that empower domain experts to quickly and efficiently populate the domain knowledge and pedagogical strategies within the ITS. To this end, creating usable and efficient ITS authoring tools can be framed as a software engineering problem. Since the design and implementation of the authoring tool directly impacts the design and implementation of the ITS (and vice versa), the authoring tool must be considered at the beginning of the project and developed in concert with the ITS as opposed to being developed near the end of a project and constrained to work within an existing ITS implementation. In the following subsections, we will enumerate software principles and features that should be considered for inclusion in a domain expert-centered ITS authoring tool.

2.1 Adopt a Familiar User Interface Paradigm

From a usability standpoint, the most important feature of an authoring tool is its user interface (UI). Ideally, an ITS authoring tool should present a UI that is familiar and intuitive for the type of domain expert who is intended to use it. Instead of requiring the domain expert to conform to unfamiliar ITS naming conventions and authoring workflow, the authoring tool should be modeled after software that the domain 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 domain expert is a computer science professor, 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 ITS authoring tool, modeling after an existing tool leverages decades of real-world usability and efficiency improvements.

Modeling an ITS authoring tool's UI after an existing authoring tool, like Microsoft PowerPoint, does not imply that the ITS content must be as simple as the content in a typical PowerPoint presentation. This would indeed be challenging since an ITS is likely to require authoring of complex pedagogical strategies or annotation of incorrect and correct answers which is not afforded by the PowerPoint UI. Instead, this 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 ITS authoring tool might represent blocks of curriculum knowledge as "slides" in a PowerPoint-like authoring tool. 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 proficiency. The slide could also be associated with editable "tags" that represent ITS-specific metadata such as the skills or concepts represented by the slide. Without the domain expert explicitly authoring it, the ITS could use this metadata and the student model to determine the next slide to display to the student without the domain expert explicitly authoring every possible sequence.

2.2 Include Standard Editing Features

Modeling an ITS 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 no time to refactor data models or add revision tracking.

2.3 Support Author Collaboration

An ITS authoring tool should implement features that allow multiple domain 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 ITS 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 ITS 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 ITS and the authoring tool since it would impact the architecture and implementation of the entire system.

2.4 Facilitate Rapid Iteration and Testing

To facilitate refining of the curriculum content or ITS behavior, 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 domain expert can ideally interact with the ITS while editing content in real-time or with only a minor delay. This feature allows the domain expert to quickly confirm that content is presented in a visually appealing manner in the ITS and that the tutor behaves correctly while the domain expert is modifying properties or settings that influence the ITS 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 ITS data models would be required to support 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 ITS depending on the types of edits the domain expert is making. Revision tracking of data changes previously mentioned in the description of the collaboration feature would also be useful in implementing a rapid iteration feature.

2.5 Accommodate Novice and Expert Authors

The ITS authoring tool should support editing methods that are specifically tailored to novice and expert users rather than presenting a one-size-fits-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 as opposed to providing direct access to advanced settings and authoring mechanisms. Therefore, for less frequently used authoring activities or when authoring complex knowledge representations or ITS-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 may ask the domain expert a series of questions that are used to generate a set of rules for grading the answer. On the other hand, an expert user would have the option of bypassing the wizard and authoring the rules directly. Interestingly, this feature could be supported by embedding an ITS in the authoring tool itself to assist the domain expert in authoring content.

2.6 Automation of Complex Tasks

Some aspects of authoring domain knowledge or ITS behavior may be too complicated or too labor intensive for a domain expert to accomplish manually using an ITS authoring tool. In these situations, the authoring tool should provide automated mechanisms for generating curriculum content or pedagogical strategies. For example, consider the use case for generating curriculum content for a performance-oriented ITS where students are presented with many variations of similar problems while the ITS provides step-by-step tutoring. Cognitive Tutor Authoring Tools (CTAT) simplifies the authoring of these types of problems while creating example-tracing tutors [7]. CTAT allows the domain expert to demonstrate correct and incorrect behavior while solving a problem to create a generalized behavior graph. The CTAT "mass production" feature is then used to create multiple problems that can be solved using the generalized behavior graph. This gives the example-tutor the ability to tutor students for problems that were not manually annotated by a domain expert using an authoring tool, while not requiring the author to understand the computational models needed to generate the new content.

Another approach to simplify the authoring of pedagogical strategies is to assist the domain expert though the use of data mining techniques. Instead of authoring an ITS with pedagogical strategies for every possible 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, a web-based learning environment was data mined to inform teachers of students who were at risk [10]. 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 relatively little pedagogical scaffolding. After collecting student answers, the data could be mined to identify common misconceptions or domain knowledge that may require additional scaffolding by the ITS. The ITS authoring tool would flag sections of the domain knowledge or identify broader concepts that the domain expert could then focus on improving. This would naturally lead to an iterative authoring process where the ITS continues to evolve by focusing effort on the issues most relevant to students who are using the ITS. Using this type of ITS authoring assistance feature has the potential to dramatically reduce the amount of authoring effort since the domain expert is not required to exhaustively predict and annotate all possible correct and incorrect answers. On the other hand, the initial iterations of the ITS are not likely to be particularly effective since it will have limited ability to remediate students who are having difficulty.

3 Lessons Learned from the LEONARDO Digital Science Notebook

For the past three 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 a web browser on Windows and Mac OS X computing platforms. LEONARDO integrates intelligent tutoring systems technologies 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 pedagogical agent, supports students' learning with real-time problem-solving advice. LEONARDO's curriculum is based on that of the Full Option Science System [11]. Throughout the inquiry process, students using the LEONARDO CyberPad are invited to answer multiple-choice questions, write responses 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. Domain experts use the Composer (Figure 1) authoring tool to create curriculum content displayed in the digital science notebook as well as rules and dialogue 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 of a domain expert-centered authoring tool (discussed in the previous section) were "discovered" as required features or features that would enhance the productivity of domain experts.

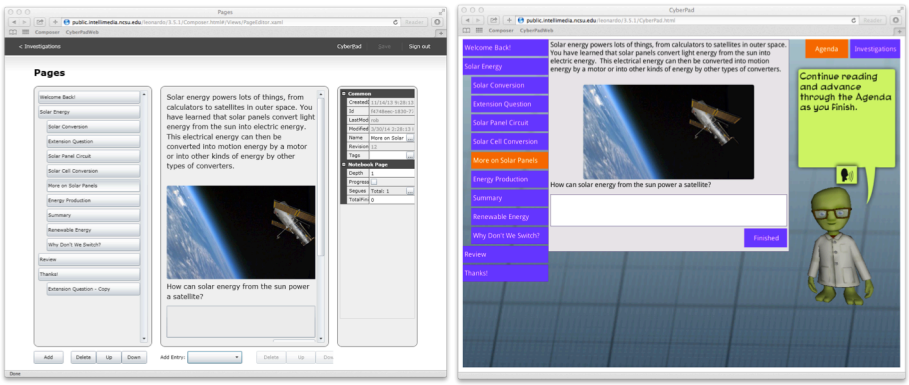

Figure 1. The Composer tool (left) and CyberPad (right) in rapid iteration editing mode

The LEONARDO project did not originally include an ITS 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, domain 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 authoring domain content was highly inefficient. It was labor intensive and error prone due to manually copying data. In addition, pedagogical agent rules and dialogue were tightly coupled with the contents of the XML documents making the entire system brittle and easily broken by syntax and typographical errors in the XML documents.

This initial approach to ITS authoring for the LEONARDO project had several significant drawbacks: First, the domain experts did not have a means to visualize what the curriculum content and 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 domain 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 often 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 [6].

Based on this initial authoring experience and future plans to dramatically increase the amount of curriculum content and pedagogical agent dialogue, it became an 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 domain experts who would use the tool in the future: elementary school teachers, college of education graduate students, and faculty. We then proceeded to design Composer's UI by reviewing authoring tools from other domains that our domain 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 the previous section.

The Composer authoring tool dramatically improved the authoring workflow for the LEONARDO project in years two and three. Domain experts were empowered to author and refine curriculum content and pedagogical agent rules independently of the development team. In addition, a familiar workflow and features such as rapid iteration, copy, cut, and paste further improved the efficiency of domain experts. These improvements did come at a development cost of refactoring data models, logic, and storage to make it possible to possible to edit and track small discrete parts of the curriculum independent of the rest of the curriculum data.

4 Conclusions

Widespread development and deployment of ITSs depends on efficient transfer of domain knowledge and pedagogical strategies from domain experts to the ITS. Authoring tools hold great promise to facilitate knowledge engineering. However, it

should be emphasized that authoring tools should be tailored to the domain 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 automation features to assist in the authoring of pedagogical strategies and to identify parts of the curriculum that need additional scaffolding as indicated by mining student data. Likewise, it will be important to design and implement novice and expert UIs to simplify authoring complex knowledge and underlying ITS mechanisms.

Acknowledgments. The authors wish to thank members of the IntelliMedia Group for their assistance. This work was supported by the National Science Foundation under Grant DRL-1020229. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

5 References

1. Woolf, B.P.: Building Intelligent Interactive Tutors: Student-centered strategies for revolutionizing e-learning. (2009).
2. Nwana, H.: Intelligent tutoring systems: an overview. Artif. Intell. Rev. 4, 251–277 (1990).
3. Beal, C.R., Walles, R., Arroyo, I., Woolf, B.P.: On-line tutoring for math achievement testing: A controlled evaluation. J. Interact. Online Learn. 6, 43–55 (2007).
4. Graesser, a. C., Chipman, P., Haynes, B.C., Olney, A.: AutoTutor: An Intelligent Tutoring System With Mixed-Initiative Dialogue. IEEE Trans. Educ. 48, 612–618 (2005).
5. VanLehn, K.: The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. Educ. Psychol. 46, 197–221 (2011).
6. Murray, T.: An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. In: Murray, T., Blessing, S., and Ainsworth, S. (eds.) Authoring Tools for Advanced Technology Learning Environments. pp. 493–546 (2003).
7. Aleven, V., Mclaren, B.M., Sewall, J., Koedinger, K.R.: A New Paradigm for Intelligent Tutoring Systems : Example-Tracing Tutors. Int. J. Artif. Intell. Educ. 19, 105–154 (2009).
8. Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., McGuigan, N.: ASPIRE : An Authoring System and Deployment Environment for Constraint-Based Tutors. Int. J. Artif. Intell. Educ. 19, 155–188 (2009).
9. Nkambou, R., Bourdeau, J., Psyché, V.: Building Intelligent Tutoring Systems: An Overview. In: Nkambou, R., Bourdeau, J., and Mizoguchi, R. (eds.) Advances in Intelligent Tutoring Systems. pp. 361–375 (2010).
10. Merceron, A., Yacef, K.: Educational Data Mining: a Case Study. AIED. pp. 467–474 (2005).
11. Mangrubang, F.R.: Preparing elementary education majors to teach science using an inquiry-based approach: The Full Option Science System. Am. Ann. Deaf. 149, 290–303 (2004).