PUMPT: an e-Textbook platform based on a Personal User Model for Learning

Judy Kay[1 [0000−0001−6728−2768]] and Bob Kummerfeld[1 [0000−0002−6046−6393]]

School of Computer Science, University of Sydney, Australia

Abstract. While classic textbooks are increasingly available online, future e-textbooks have the potential to fit within a rich ecosystem of electronic learning tools commonly used in schools and universities. We describe work towards that vision, based on the *Platform for User Model Personalized Textbooks* (PUMPT) platform. This integrates learning data from multiple e-textbook and other learning tools, notably a learner management system with its data from class assignments, quizzes and exams. All such data can be used to provide an Open Learner Model (OLM) to support learners in self-monitoring, reflection and planning that is based on a holistic view. Our core contribution is the new PUMPT platform for e-textbooks that are integrated with other learning tools. [1]

Keywords: e-textbooks · personalization · user models

1 Introduction and related work

The simplest form of e-textbook is like a paper textbook, with the same content, but with valuable affordances, such as easy, fast and powerful searching, highlighting, bookmarks, translation and easily accessed links to external sources [21]. Beyond this, many e-textbooks integrate rich and interactive content, such as visualisations, tests and surveys [8, 22, 9]. Notable examples come from computer science education [17, 7, 5, 19, 22], with demonstrated learning benefits, for engagement, motivation and performance [22]. Some e-textbook work has also integrated learner modelling [10, 2] to harnesses the data from the learner's use of the e-textbook.

We want to go beyond this, with learning data from the e-textbook integrated with data from other learning tools that students use in typical university subjects. Core to our approach is a long term user model and its associated Open Learner Model (OLM) [4], with their demonstrated benefits [3].

Most work on personalised teaching systems has a learner model as part of the system [23, 20]. By contrast, our learner model is decoupled from the e-textbook and is part of the user's personally-controlled Personalised User Model for Learning (PUML) [15] as a unified store of learning data. This paper presents PUMPT, and the PUML, which is based on the Personis user modelling system [12, 1, 14].

[1] Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-2895/paper12.pdf

2 Architecture

Figure 1 shows how our e-textbook platform, and the e-textbooks authored on it, fit into a typical university student's learning in a semester. As with conventional textbooks, we envisage that each of blue e-textbooks at the top has all been written by teachers who draw on their experience to create a rich set of content that they used successfully in their own teaching. The academic who designs each university subject can select an e-textbook and use it in various ways as part of their teaching. Importantly,the academic also makes use of other teaching tools, such as the near universal LMSs which typically have learning materials, various tools such as discussion boards, and a grade-book.

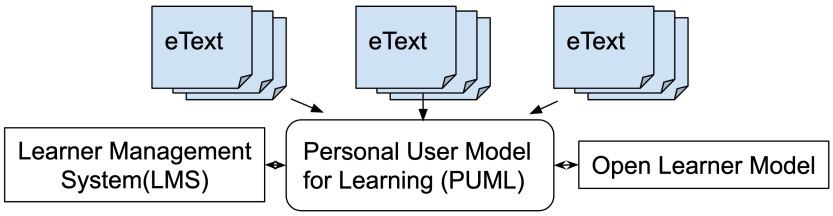

Fig. 1. Learner view of their e-textbooks, the Learner Management System, and their Personal User Model for Learning (PUML), with an OLM interface.

The figure shows how the e-textbooks and other learning technology, such as the LMS, can make use of a Personal User Model for Learning (PUML) [15]. This is a long term store of information about the learner, organised within a user model. We keep the PUML in a Dropbox file system that has been mapped into the user file space. This is a return to the roots of our approach [11] where the user model was kept in the user's filespace on a unix system − in later work we moved to user model servers [12, 13].) This meets our goal for creating a PUML that is ultimately the control of the learner.

At the same time this approach ensures the PUML is stored securely on a cloud storage system. This reduces the risks of data loss associated with pure local storage (for example, due to data corruption from device faults, change of technology and user error). Dropbox provides very high reliability and security as well as insulating a user from regular maintenance and upgrades. Importantly, it also means that the PUML can be available to the learner on any personal device with access to the PUML in the Dropbox.

We now turn to the detailed architecture of our PUMPT platform. Figure 2 shows the authored e-textbook content at the left − this has the typical elements of e-textbooks, expository content, interactive activities (e.g., multiple choice questions, fill in the blank questions etc).

As the web browser runs locally on the learner's machine, all personalisation and interactivity occurs locally to produce the interface the learner sees. It sends

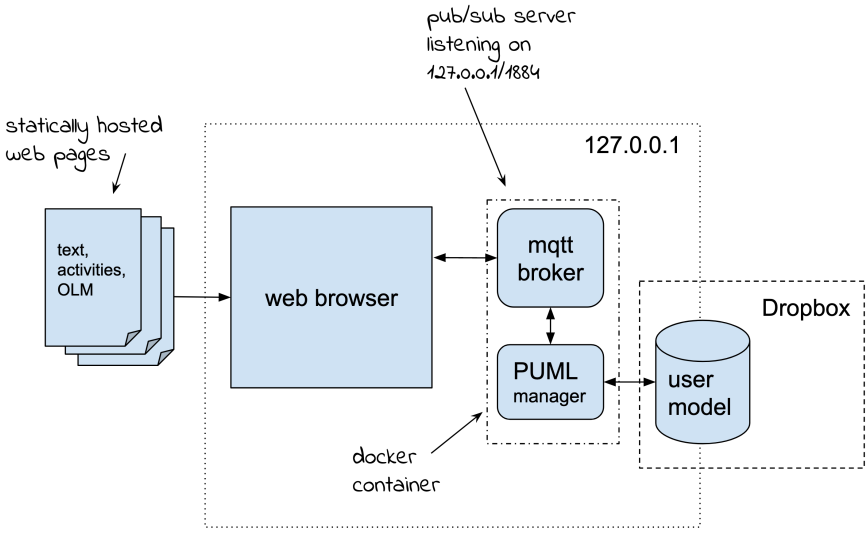

Fig. 2. Architecture of the PUMPT e-textbook platform and the PUML

data from the interaction, including information about page views, to the PUML. The user model in the PUML is available to the learner via the OLM. The model in the PUML is also available to personalise the e-text experience.

We now drill down to the technical details that make this possible. The e-textbook is implemented as a set of web pages generated from an extended version of the "reStructured Text"[2] markup language. A preprocessor scans the input files in the markup language and converts the extended directives into raw HTML, passing the normal reStructured Text markup through to the input files that the Sphinx[3] document processor uses to generate the final HTML pages. The extended directives that the textbook author can use include:

- page meta data: used to report when a user views the page;
- fitb: learner model evidence from fill-in-the-blank questions;
- mcq: learner model evidence from multiple choice questions;
- likert: learner model evidence from answers to Likert-scale questions;
- OLM: display the open learner model;
- log: other logging created by the etext author.
- showhide: show or hide content based on the user model

These are converted to HTML/Javascript which is processed in the browser during user interaction.

Personalisation of the e-text currently takes a very simple approach. Authors write the e-textbook content as "reStructured Text" markup text. They can use

[2] https://docutils.sourceforge.io/docs/user/rst/quickref.html
[3] https://www.sphinx-doc.org/en/master/

our extended directive, showhide, to mark up text that is conditionally presented, depending on the value of a component in the user model. This supports selective presentation of content, including navigation links. This is similar to our early work [16] and we demonstrated that this simplicity can also support scrutability of the personalisation [6, 14]. Another extension allows inline content to be modified with content from the user model.

The personalisation process that generates what an individual learner sees is controlled by the Personis user modelling system. As the learner loads and interacts with each page of the e-text, the markup directives and the learner model drive the personalisation. We are well aware that this personalisation process imposes a load on the e-text author. They need to add the relevant directives in the markup for the e-textbook. The textbook author also needs to define the learner model by specifying the learner model ontology which has:

- contexts, which form a hierarchy − for example, an e-text on the programming language, C, may have contexts for high level aspects such as *control structures* and *data structures* and within these there may be more fine-grained contexts, such as *loops* and *selection* within *control structures*;
- components which are the leaves of the ontology hierarchy − for example, *for-loops* and *dangling-else* within the context C/control structures/loops;
- each component can have *rules* that are examined when each "tell" adds evidence to the component. Rules may add evidence to other components or send data to external applications;
- each component has one or more *resolver* functions that interpret the available evidence to return the value of the component.

Depending on the nature of the textbook, the design of this ontology may be reasonably straightforward for the author as they design the goals of the e-textbook. Essentially, the design aims to make the simplest personalisation simple to implement, and more sophisticated forms possible with more effort.

A key part of the system is the publish/subscribe[4] paradigm for data handling. As the user interacts with the e-text pages, messages are "published" to "topics" on a local pub/sub message broker. The PUML manager subscribes to these topics, and then receives each message the PUMPT publishes. These update the user model (using the Personis "tell" operation). This is a very simple and elegant approach to managing data from user interaction as it avoids explicit handling of asynchronous server calls. It is also very flexible and extensible as there can be more than one subscriber to topics. The publish/subscribe message broker used in the system is Eclipse-paho mosquitto[5]. This program is compact (<0.25M) and efficient, easily capable of handling the modest message load of a single user. It allows raw TCP connections as well as a web-sockets (a thin protocol layer over TCP used by web applications). Mosquitto is available for Linux, MacOS and Windows.

[4] `https://en.wikipedia.org/wiki/Publish-subscribe_pattern`
[5] `https://mosquitto.org/`

Notably, there is no central server for managing user models. Each learner runs their own copy of the MQTT message broker and the PUML system and accesses the PUML data in Dropbox (or other cloud storage). Both these components (broker and PUML) run in a docker container[6] for ease of installation and isolation. This removes the need for authentication in order to access the PUML since the docker container is running purely locally on the learner's desktop or laptop and is not accessible from the outside network. The model data is stored in Dropbox which requires separate user authentication and is private to the user.

Web pages in the e-text contain the Eclipse-paho MQTT javascript library[7] that allows web-socket client connections to the MQTT broker. When a user interacts with the page, for example to do a Multiple Choice Question, messages containing the user response are published to the local broker (using the "localhost" address). These messages are then processed by the PUML manager and the model updated with evidence from the messages.

Inside the PUML, as new evidence is received, Personis resolvers and rules[1] interpret the available evidence and determine the current value of components and generate a change request in the Open Learner Model display. These requests are published by Personis to relevant topics on the pub/sub broker. The OLM page has javascript subscriptions to these topics and so will receive the requests and update the OLM display accordingly.

A PUML is an example of a Personal Data Store[8]. From wikipedia: "Personal data services or personal data stores (PDS) are services to let an individual store, manage and deploy their key personal data in a highly secure and structured way. They give the user a central point of control for their personal information (e.g. interests, contact information, affiliations, preferences, friends). "

In our case the Personal Data Store contains learning data gathered from user interactions with the e-text. However, the data contained in the PUML could come from many sources, not just the e-text. It could, for example, be collected from multiple e-texts and gather evidence for common learning objects. It could be data from a learner management system (eg Canvas) and include interaction events and test scores.

3 Summary and conclusions

We have presented an overview of the architecture of PUMPT, our e-textbook platform. We developed PUMPT because as authors and teachers, we wanted to create e-textbooks and could not find a platform with the features we wanted. We now briefly describe them. This paper relates mainly to one of our goals - to enable learners to control their own learning data and its use. Also, we wanted a platform that made authoring manageable in the sense that the technical burden of creating the learning materials was acceptable. At the same time,

[6] `https://www.docker.com/`
[7] `https://www.eclipse.org/paho/index.php?page=clients/js/index.php`
[8] `https://en.wikipedia.org/wiki/Personal_data_service`

we wanted the platform to provide flexibility in authoring and in creating new tools for authoring expository content and self-assessments. For these aspects, we settled on use of a markup language (even though a WYSIWYG interface would be nicer and this is on our roadmap). A key goal was to ensure that we could create the e-textbook with an Open Learner Model to support the learner in key metacognitive processes of self-monitoring, reflection and planning their learning. The creation of OLM interfaces is a current project.

We have also been deeply aware of the authoring challenges of even a conventional textbook. This paper has described the personalisation mechanisms we have created − these are the most basic approach of marking material for optional presentation to the learner. This will create a significant additional content authoring burden on teachers. At the same time it does provide enough flexibility for some very valuable forms of personalisation that we consider important. We note that the use of the term, personalisation, is used in many ways in educational contexts. For example, if students are allowed to proceed at their own pace, even using paper learning materials, this is a valuable form of personalisation. In our PUMPT roadmap, this form should be supported. For example, we would like to enable a learner to use their OLM to monitor their progress. But we also consider it critical to enable them to do this in relation to a meaningful standard. A very simple one could be created by the teacher who is using the textbook − they could create a timeline for progress so that this could be presented in the OLM so the student could see their progress and compare it with the teacher's expectations. Supporting this form of personalisation would require a simple way for the teacher who is using the e-textbook to create such a timeline. This is part of a larger vision for supporting personalisation that places a layer of information over the textbook. For example, a typical textbook has several chapters and the teacher using it may want to distinguish chapters (or parts of them) that are:

- *core* to the subject and should be the top learning priority for students;
- only relevant for students who want to do *advanced material* (and should only be tackled after the core has been mastered);
- not part of the subject and can be skipped.

There are other forms of personalisation, which have recently been distinguished as customisation, individualisation and adaptation [18]. Each of these calls for creation of additional material which can be selectively presented to the learner. Our current approach can support simple forms of these.

A distinctive driver for the design of this architecture has been to provide personalisation that is driven by a learner controlled Personal User Model for Learning (PUML). This builds on our long term work on the Personis user modelling framework that aims to ensure that teaching system frameworks and learning materials are designed, from their foundations to enable the user to scrutinise and control their own learner model and its use.

References

[1] Mark Assad, David J. Carmichael, Judy Kay, and Bob Kummerfeld. "PersonisAD: distributed, active, scrutable model framework for context-aware services". In: *Proceedings of pervasive 07, 5th international conference on pervasive computing*. Vol. 4480. LNCS. Springer, 2007, pp. 55–72. ISBN: 978-3-540-72036-2. DOI: http://dx.doi.org/10.1007/978-3-540-72037-9_4.

[2] Ivica Boticki, Gökhan Akçapınar, and Hiroaki Ogata. "E-book user modelling through learning analytics: the case of learner engagement and reading styles". In: *Interactive Learning Environments* 27.5-6 (2019), pp. 754–765.

[3] Susan Bull. "There are open learner models about!" In: *IEEE Transactions on Learning Technologies* 13.2 (2020), pp. 425–448.

[4] Susan Bull and Judy Kay. "SMILI: a framework for interfaces to learning data in open learner models, learning analytics and related fields". In: *I. J. Artificial Intelligence in Education* 26.1 (2016), pp. 293–331. DOI: 10.1007/s40593-015-0090-8. URL: https://doi.org/10.1007/s40593-015-0090-8.

[5] Phillip Compeau and Pavel A Pevzner. "Life after MOOCS". In: *Communications of the ACM* 58.10 (2015), pp. 41–44.

[6] Marek Czarkowski and Judy Kay. "A scrutable adaptive hypertext". In: *AH '02: Proceedings of the second international conference on adaptive hypermedia and adaptive web-based systems*. Springer-Verlag, 2002, pp. 384–387.

[7] Barbara J Ericson, Mark J Guzdial, and Briana B Morrison. "Analysis of interactive features designed to enhance learning in an ebook". In: *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*. 2015, pp. 169–178.

[8] Barbara J Ericson and Bradley N Miller. "Runestone: A Platform for Free, On-line, and Interactive Ebooks". In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 2020, pp. 1012–1018.

[9] Eric Fouh, Ville Karavirta, Daniel A Breakiron, Sally Hamouda, Simin Hall, Thomas L Naps, and Clifford A Shaffer. "Design and architecture of an interactive eTextbook–The OpenDSA system". In: *Science of computer programming* 88 (2014), pp. 22–40.

[10] Yun Huang, Michael Yudelson, Shuguang Han, Daqing He, and Peter Brusilovsky. "A framework for dynamic knowledge modeling in textbook-based learning". In: *Proceedings of the 2016 conference on user modeling adaptation and personalization*. 2016, pp. 141–150.

[11] J. Kay. "The um toolkit for cooperative user modelling". In: *User Modeling and User-Adapted Interaction* 4.3 (1995), pp. 149–196. ISSN: 0924-1868.

[12] J. Kay, B. Kummerfeld, and P. Lauder. "Personis: a server for user models". In: *Proceedings of AH 2002, 2nd international conference on adaptive hypermedia and adaptive web-based systems*. Ed. by P. De Bra, P.

Brusilovsky, and R. Conejo. Vol. 2347. LNCS. Springer, 2002, pp. 203–212.

[13] Judy Kay and Bob Kummerfeld. "Creating personalised systems that people can scrutinise and control: drivers, principles and experience". In: *ACM Transactions on Interactive Intelligent Systems (TiiS), Highlights of the Decade in Interactive Intelligent Systems* 2.4 (2013), p. 24.

[14] Judy Kay and Bob Kummerfeld. "Creating personalized systems that people can scrutinize and control: Drivers, principles and experience". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 2.4 (2013), pp. 1–42.

[15] Judy Kay and Bob Kummerfeld. "From data to personal user models for life-long, life-wide learners". In: *British Journal of Educational Technology* 50.6 (2019), pp. 2871–2884.

[16] Judy Kay and RJ Kummerfeld. "An individualised course for the C programming language". In: *Proceedings of second international WWW conference*. 1994, pp. 17–20.

[17] Ari Korhonen, Thomas Naps, Charles Boisvert, Pilu Crescenzi, Ville Karavirta, Linda Mannila, Bradley Miller, Briana Morrison, Susan H Rodger, Rocky Ross, et al. "Requirements and design strategies for open source interactive computer science ebooks". In: *Proceedings of the ITiCSE working group reports conference on Innovation and technology in computer science education-working group reports*. 2013, pp. 53–72.

[18] Natalia Kucirkova, Libby Gerard, and Marcia C Linn. "Designing personalised instruction: A research and design framework". In: *British Journal of Educational Technology* ().

[19] Kerttu Pollari-Malmi, Julio Guerra, Peter Brusilovsky, Lauri Malmi, and Teemu Sirkiä. "On the value of using an interactive electronic textbook in an introductory programming course". In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. 2017, pp. 168–172.

[20] John Self. "The defining characteristics of intelligent tutoring systems research: ITSs care, precisely". In: *International journal of artificial intelligence in education* 10.3-4 (1999), pp. 350–364.

[21] Kimberly Anne Sheen and Yan Luximon. "Student perceptions on future components of electronic textbook design". In: *Journal of Computers in Education* 4.4 (2017), pp. 371–393.

[22] David H Smith IV, Qiang Hao, Christopher D Hundhausen, Filip Jagodzinski, Josh Myers-Dean, and Kira Jaeger. "Towards Modeling Student Engagement with Interactive Computing Textbooks: An Empirical Study". In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 2021, pp. 914–920.

[23] Beverly Park Woolf. *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann, 2010.