MOOClm: User Modelling for MOOCs

Ronny Cook(✉), Judy Kay, and Bob Kummerfeld

School of Information Technology, University of Sydney,
Sydney, NSW 2006, Australia
{ronald.cook,judy.kay,bob.kummerfeld}@sydney.edu.au
http://chai.it.usyd.edu.au/

Abstract. Emerging MOOC platforms capture huge amounts of learner data. This paper presents our MOOClm platform, for transforming data from MOOCs into independent learner models that can drive personalisation and support reuse of the learner model, for example in an Open Learner Model (OLM). We describe the MOOClm architecture and demonstrate how we have used it to build OLMs.

Keywords: MOOCs · Learner modelling · Open Learner Modelling (OLM) · Learner model server

1 Introduction and Background

MOOCs (Massively Open Online Courses) are based on platforms designed for teaching on a massive scale. These can support SPOCs, Small Private Online Courses [Fox, 2013] and MOOClets [Williams, 2014] which teach one topic. These platforms log learner activity extensively. Our MOOClm platform has been designed to harness this data by transforming it into an independent learner model. This means that the model can be reused in other learning systems. We illustrate one such use, in an Open Learner Model (OLM) interface [Bull and Kay, 2010]. This offers promise of the demonstrated benefits of OLMs [Bull and Kay, 2013; Mitrović and Martin, 2002]. It may also help address the problem of high dropout rates in MOOCs [Kizilcec et al., 2013]. Drawing on studies of learner preferences for OLMs [Bull, 2012] and the potential to support metacognitive processes [Bull and Kay, 2013], we designed the learner model and OLM to enable a learner (or other stakeholder, such as a mentor or teacher) answer the following questions:

1. **Overview:** What is the overall progress of this student on the learning activities?
2. **On-track:** In which learning objectives has the learner met the teacher expectations?
3. **Behind:** In which learning objectives are they lagging behind expectations?
4. **Activity-Type-Progress:** What are the answers to Q2-3 for a particular class of activity (video, exercise, discussions....)
5. **Act:** How can the student find learning resources associated with any given learning outcome?

F. Ricci et al. (Eds.): UMAP 2015, LNCS 9146, pp. 80–91, 2015.
DOI: 10.1007/978-3-319-20267-9_7

Current MOOCs store detailed logs for use of learning resources and performance on assessment tasks. There has been work towards standards for such data, for example MOOCdb [Veeramachaneni et al., 2013] and the eXperience API (aka Tin Can) [Mueller et al., 2014]. A recent review of analytics for major MOOC platforms [Muñoz-Merino et al., 2015] concludes they are rudimentary and they do not address our questions. Some MOOCs have analytics that partially answer Q1 and 4; for example Khan Academy's math course has a skillometer for fine-grained skills and badges for coarse grained ones. MOOClm moves beyond these to create a learner model of the course *learning objectives*. This is an independent learner model server [Brusilovsky et al., 2005; Kay et al., 2002].

2 Related Work

MOOCs (Massively Open Online Courses) are defined by the goal to deliver high quality tertiary level courses to thousands of students, at low per-student cost. They are characterised by online video lectures for delivery of content and support for low cost formative assessment based on self-assessment exercises, online discussion forums and peer review tools [Breslow et al., 2013; Kay et al., 2013]. MOOC platforms have also been used for SPOCs (Small Private Online Courses) [Fox, 2013] where they have been used very effectively for blended learning [Waldrop, 2014].

High budget MOOCs draw upon considerable expertise in learning design. This makes use of careful curriculum design, including careful definition of the learning objectives. Some MOOCs, notably some Khan Academy courses, make the learning objectives and individual progress in them available to the learner, in forms of skill-meter like displays and badges [Thompson, 2011]. However, there has been no report of a systematic approach to curriculum mapping to define MOOC curricula and inform the design of the MOOC. This is in stark contrast to the widespread practice in K-12 education [Jacobs, 1997, 2004] where the curriculum as seen as a work-in-progress, with the curriculum designer refining it to ensure that the actual learning materials match the intended learning goals.

For the purposes of this mapping, we define *learning objectives* as what the curriculum is intended to teach and we link these to the *learning objects*. Curriculum mapping distinguishes those learning objectives that are *taught* (as in video lectures) and those that are *assessed*. The latter are critical for evaluating the effectiveness of the curriculum as they can provide evidence of the learning outcomes actually achieved by learners. Gluga's ProGoSs [Gluga et al., 2012, 2013] provided a platform for systematic curriculum mapping, and it completes the loop by linking summative assessment data into the system.

Open Learner Models (OLMs) make the system's model of the learner knowledge and characteristics available to the student [Bull and Kay, 2010]. They vary markedly, with their design driven by the particular purposes of the OLM [Bull and Kay, 2007]. They have been primarily designed for use by learners, to support reflection, planning and seeing progress. A key challenge for the design of an OLM is to create a suitable interface. Some key examples are INGRID

[Conejo et al., 2012] a public tool for visualising learner models and the extensive work of Susan Bull and colleagues, such as their "OLMlets" [Bull et al., 2008]. Our work aims to create a OLM, and associated underlying learner model, that is for use by the MOOC curriculum designer.

3 MOOClm Architecture for the Open edX Platform

Figure 1 gives an overview of the MOOClm architecture. At the left is the MOOC platform, the Open edX platform. This is widely used and open source. This was the reason it was chosen for our SPOC on *C and Unix*. The right of the figure shows the MOOClm server. We now explain its design, in terms of the underlying ontology and the approach to representing the elements needed for a OLM that could enable learners to answer the questions described above.

A key first step in designing the learner model was to create the ontology, or namespace, for the model. To make this systematic, we build upon the standard learning design practice which defines the *intended learning outcomes* for a course. These form a hierarchy, with more general learning objectives being refined as a set of detailed learning objectives.

As our SPOC was a core area of computer science, we started the process of defining the learning objectives by drawing upon those defined in the ACM CS2013 curriculum [ACM Joint Task Force, 2013]. Our approach is similar to ProGoSs [Gluga et al., 2012] with its interface for teachers to define their course in terms of its intended learning objectives, taking foundations from a standard curriculum. This also similar to the approach in [Apted et al., 2004] where an authoritative resource, such as an online dictionary, can be mined to create a base set of key terms that can be used as learning objectives. It is desirable to build from a standard set of learning objectives since that facilitates learner model reuse, across MOOCs and other learning platforms and towards a lifelong learner model [Kay, 2008].

In addition to the standard learning objectives, we found that additional objectives were needed if we were to provide useful answers to our core OLM questions. This happened also in the work of Apted et al. [Apted et al., 2004] where the teacher needed to add local specialised terms, where there were multiple synonyms in wide use, and there the course structure made use of additional concepts. In the case of MOOClm , we needed to augment the ACM CS2013 curriculum because it is lacks the fine grained detail that is relevant for a learner who needs to track their progress in the important MOOC concepts. For example, the ACM CS2013 curriculum has a single learning objective for a collection of several data structures; but the focus of key lectures and exercises in our SPOC were specifically about C linked lists. In addition, the ACM CS2013 curriculum is language agnostic but our students see the subject in terms of the language used in the teaching. To make the OLM ontology meaningful, we need to augment the ACM CS2013 curriculum with these concepts. While this compromises the potential reuse of the learner model across MOOCs/SPOCs, it is essential.

Figure 1 shows how MOOClm augments the MOOC. We will illustrate this with an example of a student 'Alice" tackling a set of learning objectives on C pointers. facilitate re-use of the model in other contexts. [Kay, 2008] She views a video and then does the first three self-test exercises in a set.

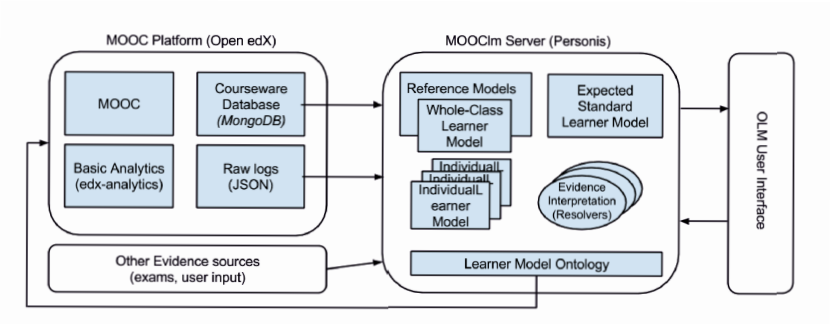

Fig. 1. MOOClm Architecture

First, consider the MOOC platform on the left of the figure. The top left box represents the MOOC interface the users see. Below this is the basic analytics tools. At the upper right, we show the MongoDB "Courseware" database of all learning resources and references, including text for exercises and YouTube references for videos. At the lower right is the raw logs of date-stamped events stored in JSON format. The most recent response for each problem is in the analytics database; the JSON logs have the full history.

The logs are comprehensive but need care to transform into a learner model. For example, when Alice views the video, this is logged as a *load_video* event when the page is opened, then several *play_video* events at two minute intervals until the video finishes. When Alice does an exercise, edX logs a *problem_check* event from browser to server, a *problem_check* event internal to the server, then a *problem_graded* request from server to browser which gives the result. Only the *problem_graded* event indicates if the submission was correct.

We illustrate some event types to indicate challenge in designing MOOClm :

play_video : occurs at a timestamped start of play, then every two minutes.
seek_video : is when the student skips back or forward while playing the video. This indicates more active interaction than *play_video*.
load_video : indicates that the static opening frame video is being shown on a page. Of negligible interest.
pause_video : indicates that the video has been paused. When play is resumed, a *play_video* event is logged.
problem_check : occurs when the learner submits a short-answer question. This event is always followed by *problem_graded*. A *problem_check* event may or may not also include whether the submitted answer is correct.

problem_graded : is an internal entry, logged when edX checks a submitted problem. Unfortunately the result is not clearly logged as correct or incorrect - instead the HTML for the generated result is logged and this must be parsed to determine the result.

load_document : is for document download - e.g. PDF the slides used in a video.

show_transcript : Show how well the student has done in the MOOC so far.

hide_transcript : Closing the performance transcript.

Others : covers numerous other event types which can be course specific. Our system logs have 474 event types, 16 native and 458 were courseware links.

After careful analysis, we concluded that a core set of events should be used for our learner model. These are *problem_graded*, *problem_check*, *play_video*, *seek_video* and *pause_video*. These capture the information needed to answer our questions. We ignore the others.

As indicated in the lower left of the diagram, evidence can also originate from external sources, such as examinations or other MOOCs. Any such evidence must also be interpreted in light of relevant learning objectives.

We now explain how the log data is treated as evidence in the learner model, on the right of the diagram. This transformation required several design decisions. This process unifies our learner model ontology of *learning objectives* with the raw MOOC data. The log entries use edX's internal courseware ID for learning materials such as video and exercises. To process the JSON log files, this ID is checked against the courseware database, mapping it to a human-readable form, for example:

```
2014-05-19T09:58:46.369909+00:00 COMP2129 RonnyCook2 problem_
check Operating Systems and Machine Principles/Week 2/C Aggregates
and Pointers/Scope quiz/Multiple Choice #2 (correct);
```

The source line from the log here is about half a page; we omit it for brevity. Pardos et al [Pardos and Kao, 2015] have more recently done useful work in standardising the edX logs as part of a larger work.

Alice's events show that she viewed the Week 2 "Pointers" video and answered the first three exercises. These events are then cross-checked against our MOOClm mapping of *learning objectives* for each learning object. Video events are recorded as evidence if part of the individual event occurs within the time interval when that learning objective was taught.

In our example, as Alice watched the introductory material explaining pointers, we transform the associated log data to a series of *play_video* evidence items in part of the learner model for C and the learning objective: *"Understands how to use the $*$, & and $->$ C operators to reference and dereference pointers"* in the C language portion of the model. The exercises *problem_graded* events also contribute to the same learning objective. A single MOOC resource may be associated with multiple learning outcomes; in this instance, the same material is associated with the learning objective: *Understands the nature and use of C pointers.*

The end result of this process is a collection of evidence events for each learning objective. The next step is to draw conclusions from this evidence about what the student knows. For this, Personis supports creation of interpretive elements called "*Resolvers*". MOOClm provides several of these. Each Resolver examines the evidence for and against for each learning outcome and draws a conclusion about the "knowledge level" of a learning objective. One very simple resolver, "Optimistic" treats a learning objective as known if there is any positive evidence for it. A more useful resolver compares the evidence for this learner against the model shown in the figure as the "expected standard". This indicates the expected evidence available for a learner who has completed all the materials that the teacher had expected for this time in the course.

In this fashion we create a set of "reference models" against which the student can compare their own performance. This set can include models belonging to other students. The specific reference models we build included:

- A complete reference model, as outlined above, populated with evidence for all recorded learner outcomes.
- A model covering all learning objectives covered by the MOOC, as an overview of the student's use of the MOOC.
- Three models covering: just the material taught in videos; just the material tested in exercises; and just the material tested in the exam. These facilitate seeing in which parts of the course the student participated, and in seeing how coverage by the different teaching and assessment mechanisms overlap.
- A model showing the material taught by the MOOC up until week 4, to check whether students had progressed beyond this point in the MOOC.
- A model covering all learning objectives in the ACM CS2013 curriculum, as a guide to the level of overlap between MOOC materials and the ACM curriculum
- a model showing all learning objectives *not* in the ACM curriculum, to check how much of students are learning is additive to ACM learner outcomes.

In service of interpreting the OLM, we built a simple interface based on the "pack" visualisation in the Javascript D3 library. This interface permits comparing different OLMs, applying a structure/visibility filter, and selection of the preferred resolver and timestamp.

The MOOC's use of the OLM is subject to some restrictions due to cross-site scripting protections in many browsers. This limits embedding of javascript into HTML sources which may wish to connect to the OLM API directly. Instead, we use a CGI script which consults the OLM before selecting appropriate text. Use of a web service is another possible approach.

The approach used here has some marked similarities to CUMULATE [Brusilovsky et al., 2005] but also several important differences. Where CUMULATE breaks down material by topic, MOOClm attempts to approach atomicity in its learning objectives, so that differently structured courses covering similar material can be compared. MOOClm integrates the evidence store with its user models rather than using a separate store, making migration easier. The Personis resolvers used by MOOClm are very similar in function to the inference agents

used by CUMULATE, but there is a greater emphasis on student feedback for the model to ensure accuracy rather than a strict reliance on observed student activities.

Some elements of our visualisation interface, particularly the overlay and filtering facilities, provide insights that can be obscured with the CUMULATE framework.

4 Validation

To validate the MOOClm framework, we implemented a learner model for a SPOC being run for a computer science course at the University of Sydney. Participation in the SPOC was voluntary; although students were expected to view all videos, access to the videos was not exclusively through the SPOC.

The SPOC consisted of 37 videos, which in turn were broken down into a total of 187 topic-specific segments, with 156 multiple-choice and short-answer self-assessment questions. We added evidence from the students' final exam, with 38 questions of which 32 were multiple-choice.

We identified 514 ties to our learning objectives in this material, with 170 distinct learning objectives assessed. The model as a whole consisted of 1105 learning objectives extracted from the ACM CS2013 curriculum, plus 171 supplementary objectives not addressed by the ACM curriculum but covered, directly or indirectly, in the SPOC course materials. Only 19 of the ACM course objectives were addressed in the course, as the ACM curriculum is largely language- and platform-agnostic whereas the material in the SPOC is specific to C and UNIX.

The SPOC had 345 participants, with 1753506 lines of edX logs collected, of which 814035 were classified as useful evidence by the criteria discussed earlier.

All examples here use the Optimistic ("any evidence is good evidence") resolver to interpret Alice's model.

Figure 2 shows a simple view of "Alice's" knowledge as represented in our OLM. This answers the first of our representative questions, giving a broad overview of the student's knowledge. Black dots represent "known" items; white dots are "unknown". The full (complete reference) ontology has been narrowed by a software filter in our viewer to show only those learner outcomes present in the MOOC; that is, it incorporates both ACM and "augmented" outcomes, but omits any outcome which is not present in the videos or self-assessment questions from the MOOC.

When Alice clicks on the circle at bottom right, she sees the *zoomed* view of the UNIX subtopic represented in figure 3. She can zoom in or out of the model freely, down to the level of individual learning objectives.

Figure 4 attempts to address our second and third questions by comparing Alice's performance against a sample benchmark for an "expected standard" student who has completed all MOOC material through to week 4. With this colour scheme, black shows items that are known and expected to be known, yellow shows outcomes that are expected to be known but which she does *not* know,

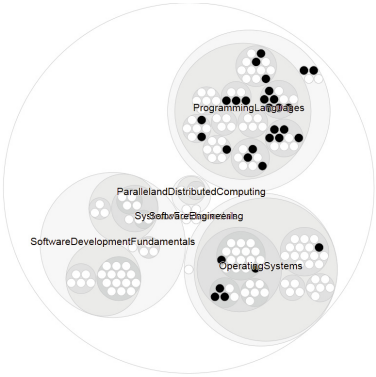

Fig. 2. Alice's OLM

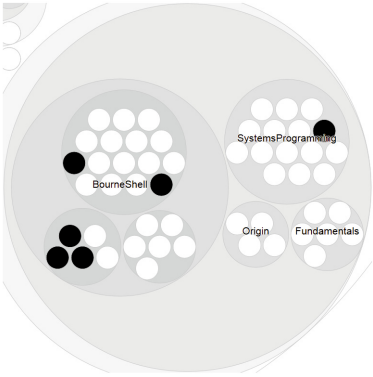

Fig. 3. Alice's OLM - Zoomed View

and the sole green dot, between the cluster of three black dots and the singleton at the bottom of the "ProgrammingLanguages" bubble, is where Alice knows something beyond the Week 4 material. Unfortunately the colour scheme does not show up well in black and white; green shows up as dark grey and yellow as very light grey.

Essentially, the yellow dots tell us where Alice is failing to meet expected performance as of week 4. The green dot tells us where she is exceeding expectations.

A similar view can be used to compare student performance against any of the reference models mentioned in section 2. Several colour schemes are available, to address colour-blindness issues and to lend emphasis to particular types of comparison.

Figure 5 answers an interesting compound question related to our earlier question 4: Of the learning objectives tested in the exam, how many have been taught by week 4 of the course? Of this set of exam-tested objectives, how well is Alice doing?

This is done by specifying a filter that only displays exam-tested results, then applying an overlay to compare our student against the set of outcomes tested by week 4 (the "week 4 curriculum").

The set involved is much smaller, since the exam can only test a small portion of the course. We see that the exam mostly tests objectives categorised under "ProgrammingLanguages", with remaining outcomes spread across "OperatingSystems" and "ParallelandDistributedComputing". The MOOC has by this time taught the material corresponding to learning outcomes for the yellow and black bubbles. Alice has learned one additional exam-related topic not covered by week 4.

In fact, comparing with figure 4 we can see that Alice has been very lucky. The former figure shows that there is a great deal of MOOC material up to week 4 of which Alice has not demonstrated knowledge. Figure 5 tells us that most of the material that Alice missed was not covered in the exam.

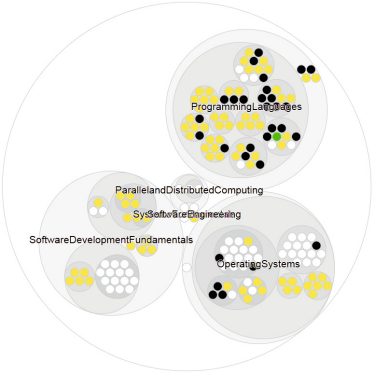

Fig. 4. Student performance against Week 4 complete reference model

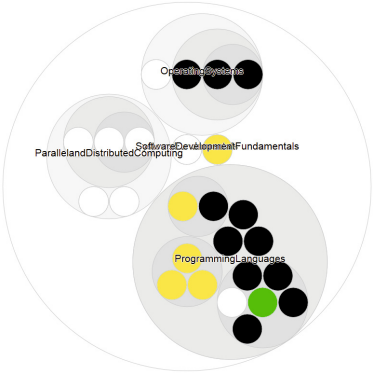

Fig. 5. Student performance against Week 4 benchmark for exam-tested learning outcomes

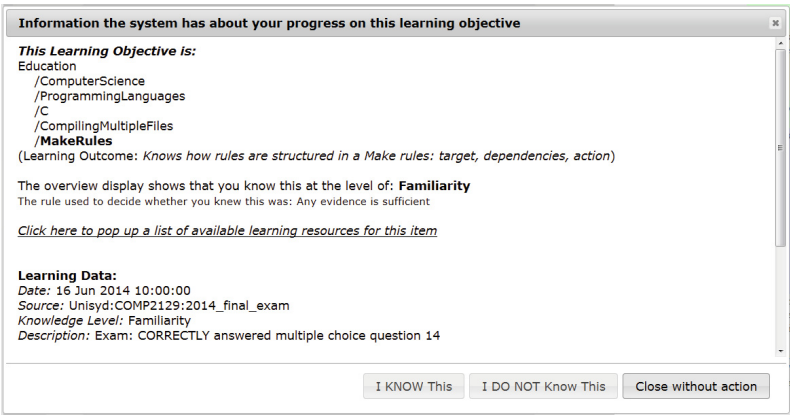

Fig. 6. Sample Evidence Snapshot

If we want to look at Alice's exam results in particular, ignoring MOOC evidence, we select a resolver that ignores MOOC evidence and only uses exam evidence. (As Alice did not participate in the final exam, the diagram showing this is quite boring.)

This viewpoint is also useful as a quick-and-dirty check that no particular part of the course was favoured in the exam. Figure 5 suggests that the UNIX-specific material may have been covered inadequately. On the other hand, it may be that the UNIX learning objectives are much more complex, so fewer individual objectives could be tested.

Figure 6 shows a snapshot of the detail for one particular learning outcome. The component's path in the model is shown, as well as a description, the resolved outcome, and a scrollable list of the evidence used to conclude this

outcome. A link is also included to a list of reference resources that the student may follow for further learning on a specified topic, directly addressing our fifth foundational question.

The interface also offers the viewer an option to override or force a value for the component, inserting an evidence item specifying the value to be resolved. This allows the student to override the value for any learning outcome that they believe to be incorrect, in principle enhancing the reliability of the model. If there is a need to exclude such explicit changes, a suitable Resolver may be used.

The same view, used against the "expected standard" model, can be used to address our fifth question. Students can view the "expected standard" model to examine evidence stored for a particular learning objective and see which learning resources will assist them with any particular learning outcome.

The interface blocks any attempt to make changes to reference models.

5 Conclusion

We have demonstrated MOOClm as a uniform framework tying together MOOCs and Open Learner models to facilitate lifelong learner models.

Integration of target models for typical and "plausibly ideal" usage gives the student a guide to their present progress, while opening the model ensures that it remains accurate. By opening the aspirational models, complete with sample evidence, we also give the learner a guide to where they should target additional learning. The MOOC may also adapt its presentation to data in the OLM, ensuring a two-way data flow.

Our contribution is the end-to-end integration of curriculum design, MOOC course construction and OLM in an integrated, open framework which facilitates re-use and lifelong learning.

In future work, we intend to adapt the framework for easier use across multiple MOOCS. The model was designed with a standardised curriculum for this reason. Use as an integrating agent across multiple MOOCs and MOOClets [Williams, 2014] will allow students to take advantage of the best parts of each platform towards lifelong learning goals. Currently, Open edX has no facility to integrate specification of learning objectives against learning objects. A critical next step is to add this capability to edX. Importantly, we need to conduct user studies to evaluate and refine the interfaces and to gain insights into the ways that learners make use of the OLM.

Acknowledgments. This research was supported by funding from the Faculty of Engineering and Information Technologies, The University of Sydney, under the Faculty Research Cluster Program. The views expressed herein are those of the authors and are not necessarily those of the Faculty.

References

ACM Joint Task Force. Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. Technical report, Association for Computing Machinery (ACM) IEEE Computer Society (2013)

Apted, T., Kay, J., Lum, A.: Supporting metadata creation with an ontology built from an extensible dictionary. In: De Bra, P.M.E., Nejdl, W. (eds.) AH 2004. LNCS, vol. 3137, pp. 4–13. Springer, Heidelberg (2004)

Breslow, L., Pritchard, D.E., DeBoer, J., Stump, G.S., Ho, A.D., Seaton, D.: Studying learning in the worldwide classroom: Research into edX's first MOOC. Research & Practice in Assessment **8**, 13–25 (2013)

Brusilovsky, P., Sosnovsky, S., Shcherbinina, O.: User modeling in a distributed E-learning architecture. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 387–391. Springer, Heidelberg (2005)

Bull, S.: Preferred Features of Open Learner Models for University Students. In: Cerri, S.A., Clancey, W.J., Papadourakis, G., Panourgia, K. (eds.) ITS 2012. LNCS, vol. 7315, pp. 411–421. Springer, Heidelberg (2012)

Bull, S., Kay, J.: Student models that invite the learner in: The smili:() open learner modelling framework. Intl Journal of Artificial Intelligence in Education **17**(2), 89–120 (2007)

Bull, S., Kay, J.: Advances in Intelligent Tutoring Systems, chapter 15: Open learner models, pp. 301–322. Springer (2010)

Bull, S., Kay, J.: Open learner models as drivers for metacognitive processes. In: Intl Handbook of Metacognition and Learning Technologies, pp. 349–365. Springer (2013)

Bull, S., Mabbott, A., Gardner, P.H., Jackson, T., Lancaster, M.J., Quigley, S.F., Childs, P.A.: Supporting interaction preferences and recognition of misconceptions with independent open learner models. In: Nejdl, W., Kay, J., Pu, P., Herder, E. (eds.) AH 2008. LNCS, vol. 5149, pp. 62–72. Springer, Heidelberg (2008)

Conejo, R., Trella, M., Cruces, I., Garcia, R.: INGRID: a web service tool for hierarchical open learner model visualization. In: Ardissono, L., Kuflik, T. (eds.) UMAP Workshops 2011. LNCS, vol. 7138, pp. 406–409. Springer, Heidelberg (2012)

Fox, A.: From MOOCs to SPOCs. Commun. ACM **56**(12), 38–40 (2013)

Gluga, R., Kay, J., Lister, R.: Progoss: mastering the curriculum. In: Proceedings of The Australian Conference on Science and Mathematics Education (formerly UniServe Science Conference) (2012)

Gluga, R., Kay, J., Lister, R., Kleitman, S.: Mastering cognitive development theory in computer science education. Comput. Sci. Educ. **23**(1), 24–57 (2013)

Jacobs, H.H.: Mapping the Big Picture. Integrating Curriculum & Assessment K-12. ERIC (1997)

Jacobs, H.H.: Getting Results with Curriculum Mapping. ERIC (2004)

Kay, J.: Lifelong learner modeling for lifelong personalized pervasive learning. IEEE Transactions on Learning Technologies **1**(4), 215–228 (2008)

Kay, J., Kummerfeld, B., Lauder, P.: Personis: a server for user models. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.) AH 2002. LNCS, vol. 2347, pp. 203–212. Springer, Heidelberg (2002)

Kay, J., Reimann, P., Diebold, E., Kummerfeld, B.: MOOCs: So many learners, so much potential. IEEE Intell. Syst. **28**(3), 70–77 (2013)

Kizilcec, R.F., Piech, C., Schneider, E.: Deconstructing disengagement: analyzing learner subpopulations in massive open online courses. In: Proceedings of the 3rd Intl Conference on LAK, pp. 170–179 (2013)

Mitrović, A., Martin, B.: Evaluating the effects of open student models on learning. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.) AH 2002. LNCS, vol. 2347, pp. 296–305. Springer, Heidelberg (2002)

Mueller, N., Dikke, D., Dahrendorf, D.: Experience API vs SCORM-how xAPI benefits technology-enhanced learning. In: EDULEARN14 Proc., pp. 1276–1284 (2014)

Muñoz-Merino, P.J., Ruipérez, J.A., Sanz Moreno, J.L., Delgado Kloos, C.: Assessment activities in MOOCs (2015)

Pardos, Z.A., Kao, K.: moocrp: An open-source analytics platform. In: Proceedings of the Second (2015) ACM Conference on Learning @ Scale, L@S 2015, pp. 103–110. ACM, New York (2015)

Thompson, C.: How khan academy is changing the rules of education. Wired Magazine, vol. 126 (2011)

Veeramachaneni, K., Dernoncourt, F., Taylor, C., Pardos, Z., O'Reilly, U.-M.: MOOCdb: developing data standards for MOOC data science. In: Proceedings of the 1st Workshop on MOOCs at the 16th Annual Conference on AIED (2013)

Waldrop, M.M.: Massive open online courses, aka MOOCs, transform higher education and science (2014)

Williams, J.J., Li, N., Kim, J., Whitehill, J., Maldonado, S., Pechenizkiy, M., Chu, L., Heffernan, N.: The MOOClet framework: Improving online education through experimentation and personalization of modules. Social Science Research Network, November 12, 2014