Haaranen, Lassi; Mariani, Giacomo; Sormunen, Peter; Lehtinen, Teemu

Complex Online Material Development in CS Courses

Published in:
Proceedings - 20th Koli Calling Conference on Computing Education Research, Koli Calling 2020

DOI:
10.1145/3428029.3428053

Published: 19/11/2020

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Haaranen, L., Mariani, G., Sormunen, P., & Lehtinen, T. (2020). Complex Online Material Development in CS Courses. In N. Falkner, & O. Seppala (Eds.), *Proceedings - 20th Koli Calling Conference on Computing Education Research, Koli Calling 2020* Article 26 ACM. https://doi.org/10.1145/3428029.3428053

Complex Online Material Development in CS Courses

Lassi Haaranen
lassi.haaranen@aalto.fi
Aalto University, Finland

Giacomo Mariani
giacomo.mariani@aalto.fi
Aalto University, Finland

Peter Sormunen
peter.sormunen@aalto.fi
Aalto University, Finland

Teemu Lehtinen
teemu.t.lehtinen@aalto.fi
Aalto University, Finland

ABSTRACT

Computer Science (CS) education has a tradition of using online materials to teach courses, whether as a part of a course or having fully online courses. Commonly, the materials to present the subject matter are not just static objects like text or images but also contain automatically assessed exercises and other interactive content. This makes the course systems inherently tied to teaching – limiting pedagogical approaches, types of exercises, and available functionality. Increasingly, CS courses utilize multiple systems to handle various learning and course management related activities. The use of multiple systems brings about traditional software engineering problems, such as development, integration, maintenance, and testing. We present two small studies (case study and interviews) to highlight the issues in developing and running modern online CS courses. Based on these two studies we argue that online courses should be developed with a stronger software engineering approach considering the development process and tools. In addition, we define the term Complex Online Learning Material (COLM) to foster discussion and further research into improving instructor practices in online education.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; *Management of computing and information systems*; • **Software and its engineering** → **Software development process management**; Software testing and debugging.

KEYWORDS

CS education, course development, online education, material development, service oriented architectures

ACM Reference Format:
Lassi Haaranen, Giacomo Mariani, Peter Sormunen, and Teemu Lehtinen. 2020. Complex Online Material Development in CS Courses. In *Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research (Koli Calling '20), November 19–22, 2020, Koli, Finland*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3428029.3428053

Koli Calling '20, November 19–22, 2020, Koli, Finland

ACM ISBN 978-1-4503-8921-1/20/11…$15.00
https://doi.org/10.1145/3428029.3428053

1 INTRODUCTION

Distance education is not a new phenomenon but the rise of Massive Open Online Courses (MOOCs) in early 2010s brought it to limelight. The possibility to complete university courses online when it best suits you, aided by increased media coverage, skyrocketed interest in online learning. Computer Science (CS) has long been in the forefront of automated assessment – partly due to automatic testing of code being inherently a CS activity. The COVID-19 pandemic of 2020 further accelerated the need and desire to teach online, CS being no exception to this.

The complexity of online courses, as well as the systems and tools used in them, have increased over the years. Many courses use multiple systems, developed as their own software projects, to achieve a variety of tasks. Interoperability protocols and extensibility of modern Learning Management Systems (LMS) allow incorporating different elements from multiple systems. However, adapting these systems for a course often requires development effort, such as implementing tests and model solutions for new exercises.

As with all software projects, increased complexity in course development presents inherent challenges like integration of systems, comprehensive testing, and incorporating developer tools. Even though we know about best practices in software development, they seem to be often forgotten when it comes to our own course development. We have noticed this on our own courses and heard about it anecdotally.

Naturally, limited time and development resources are partly to blame. The aim of the instructors is to build, for instance, high quality exercises for a course rather than spend time documenting the intricacies of the exercise system. However, we believe there are some other issues worth highlighting regarding development of such complex online learning material that is in high demand. In brief, our *main goal is to define Complex Online Learning Material (COLM) as a term and how it relates to arranging modern online courses in CS*. We do this with two preliminary studies conducted during spring 2020:

- We discuss and highlight the issues we encountered in COLM development through a case study developing our own course
- Summarize findings from focused instructor interviews regarding COLM development and online learning platforms

Our paper is structured as follows. Next, in Section 2, we discuss the relevant background regarding online courses and developing them. Section 3 defines Complex Online Learning Material as a term. Our case study is discussed in Section 4, followed by our interview study in Section 5. We conclude in Section 6 with a discussion.

2 BACKGROUND

2.1 Learning Management Systems

Learning Management Systems (LMS), also called "e-learning systems", "learning environments" and "course management systems" [6, 8, 18] among others, are web-based information systems utilized in IT-supported learning to integrate within one environment the various materials, tools and services needed in a course. Ellis [11] defined LMSs as software applications that "automate the administration, tracking, and reporting of training events". These systems can be used to integrate, for example, course assignments, lecture material, submission & grading of exercises, and discussion forums.

Some of the most utilized LMS in the world are Blackboard, Moodle, Brightspace D2L and Canvas [15]. These well known solutions have, however, presented some limitations when it comes to their utilization in Computer Sciences courses, such as the need for exercise assessment and more flexible integration with external services [16]. To answer the need of these more technically demanding courses, some institutions have developed their own systems, such as the A+ LMS developed at Aalto University [1, 16] and used in Aalto as well as other universities.

LMSs have a relevant role when it comes to the creation of online courses. Perceived usefulness and service quality of LMSs have been reported to have a significant impact on instructor satisfaction in distance learning courses [3], and LMS system quality and information quality were found to be key factors regarding instructor satisfaction in blended learning [2].

2.2 MOOCs

Starting from about 2008 criticism toward LMSs and other institutionally owned learning environments, together with the rise of open and distance learning initiatives, paved the way for the creation of more distributed and user-owned systems. The term Massive Open Online Course (MOOC) was born in the context of an open online course called Connectivism and Connective Knowledge held by the University of Manitoba in 2008, where hundreds of people participated from around the world [13]. Compared to traditional open educational resources, MOOCs focused on creation and sharing of information by participants, networking, and integration of multiple heterogeneous environments and tools, including wikis, blogs, and social networks, all happening with the facilitation of experts in the field of study [17].

Starting from 2011 a new type of MOOCs emerged, more institutionalized, structured, centralized, and less focused on sharing, networking and cooperation. These new MOOCs were still available online on platforms such as edX (https://www.edx.org) and Coursera (https://www.coursera.org), and open to everyone. They often also included machine-graded exercises. Due to their more traditional, linear structure, and focus on content consumption and exercise completion, rather than cooperation and content creation, they were more approachable for a wider audience and easier for institutions to certify. Because of the difference between the earlier MOOCs and the new more centralized MOOCs, the former came to be known as connectivist MOOCs (c-MOOCs), and the latter as xMOOCs [9]. These two approaches are so different, that according to Moe (2015) [19] "there is no theoretical or pedagogical reason" for both to share the name MOOCs.

During the last years the need to integrate both the social aspects of c-MOOCs and the organizational advantages of xMOOCs, brought to the creation of hybrid MOOCs. The hybrid approach gave positive results regarding completion rate [12, 14] and the perceived quality of education [14].

2.3 Online Computer Science Courses

When it comes to MOOCs, Computer Science topics are very popular. For example among MOOCs at HarvardX and MITx from 2012 to 2016, Computer Science courses had the highest relative number of participants [5]. In the last few years universities and other educational institutions have increased the amount of online CS courses which are available also to learners from outside the host organization. While not all online courses are officially classified as MOOC, they can still be open to a large number of participants. For example as of summer 2020 the Finnish Institute of Technology (FITech) offers sixty-four online courses from the field of Computer Sciences run by seven Finnish universities. These courses include among others topics Artificial Intelligence, Game Development, Information Security, Web Development, Programming, and Software Engineering [23]. Another relevant distinction among online CS courses, is the one between teaching offered by educational institutions and teaching offered by online platforms such as Codecademy (https://www.codecademy.com/), Pluralsight (https://www.pluralsight.com), HackerRank ((https://www.hackerrank.com), and Khan Academy (https://www.khanacademy.org).

Regarding the teaching of programming in MOOCs, there has been some concern that the tools offered by the platforms are not sufficient for proper programming education. In specific, there is a need for automated assessment of programming exercises [22], as well as tools and learning analytics to support learners' motivation, active learning techniques, and enhanced user experience [10]. In order to offer quality content and learning activities, some institutions have created their own solutions to develop and host online CS courses, such as the A+ LMS [1, 16], the ACOS server [21], and Test My Code [20].

A+ [16], the learning management system, used in our case study (see Section 4), relies on a service-oriented approach to serving online materials. The system itself handles tasks that are shared with all courses, such as user authentication, keeping track of earned exercise points, and displaying static learning content (e.g. text, images). Typically the static content is compiled from a markup language (e.g. ReStructuredText, Latex) to be displayed by A+. However, for the automatically graded exercises and other systems to support the course (e.g. student feedback, Q&A platform) external services are used. These external services rely on a variety of interoperability protocols, such as LTI [7] and the A+ protocol [16].

A+ LMS is just one example of interoperability and integrating various learning resources. The field of online CS courses is very heterogeneous, it spans a wide range of topics, learning activities and solutions regarding how the courses are developed and hosted. Often a course may rely on services from different providers, for example, to obtain and grade exercises, to collect learning analytics data, or to authenticate the users. As more services are integrated to a course the technical complexity of the learning material increases. In the following section, we outline characteristics of these more complex online courses.

3 COMPLEX ONLINE LEARNING MATERIAL

The requirement for advanced tools influences the development of new online CS courses. These courses need to integrate more traditional learning material such as text and videos, with so called Smart Learning Content (SLC) such as tools for program visualization, automatic assessment, and algorithm & program simulation [4]. The creation of such content is a complex process. In addition to creating the material, the instructors need to be able to integrate external content & services, deploy the course online, test its functionalities, and update it. As reported by Brusilovsky et al. [4], the top three difficulties in the use of SLCs were (1) finding the SLCs, (2) customize the SLCs to own needs, and (3) integrate the SLCs with the system hosting the course.

More broadly, we can consider Online Learning Activities (OLA), that might or might not be SLC [21]. These include any kind of interactive components, such as program visualizations, multiple choice questions, or any other tasks where the learner interacts with the material.

We coined the term COLM – Complex Online Learning Materials to describe the course materials that are mostly or completely online, incorporate SLC and OLA, and that are as a whole a *software project*. The written material, including images and video, is typically produced in some markup language and there is a course repository that holds the source code to produce a compiled version of the course. In some cases, such as in the case study presented in the next section, there are multiple repositories and the online material is served from multiple different systems to the student.

As software projects, COLM courses could utilize best practices from software engineering. However, this seems to be rarely true, as in our case example. Automated compiling tools could provide meaningful information regarding material compilation. Testing, continuous integration & deployment could ease managing course material and catch bugs in, for example, exercise grading early on.

3.1 Characteristics of COLM

We consider Complex Online Learning Material to be a key part of modern CS education. To put it succinctly, *COLM means that the course itself is a software project, and as such, it should be approached from a software engineering perspective*, in addition to pedagogical considerations. In summary, COLM typically has some or all of the following characteristics:

- **Material creation is software development** – The material is, at least partially, created by programming. The instructor needs both pedagogical and technical software development skills to create the course, and integrate material and tools from external sources.
- **The course itself is a software project** – The course creation and maintenance in its wholeness is a software project. Automated processes aid, or at least should aid, in the development, testing, publishing, and maintenance of the course.
- **The instructor is also a developer** – Creation of the material and integration of external resources occurs at a lower technical layer not available through existing user interfaces in a given LMS. Typically this means that there are multiple software repositories and servers to manage.
- **Integration of multiple tools** – COLM courses integrate multiple tools, possibly using different technologies (or programming languages). This is facilitated by interoperability protocols.
- **Facilitates research** – The exercises and other activities in COLM courses generate granular data for learning analytics. This data can be used to research both new pedagogical approaches as well as new technical innovations.

4 CASE IWDAP

4.1 Course Context

The Introduction to Web Development and Programming (IWDAP) course is a full online course held in its first instance from January to May 2020 by Aalto University as part of the FITech course offering. The scope of the course was to teach the basics of programming through web development, without requiring any previous knowledge in the field. The course was free of charge and the only requirement to participate was to have a Finnish social security number. The course was also held at a local high school, with the cooperation of a tutor whose task was to give weekly face-to-face guidance to the high school students taking part in the course.

The course content follows a typical pattern for web development courses, introducing HTML & CSS first and then moving on to basics of JavaScript and how to use it in the front-end. The latter half of the course is focused on HTTP and how back-end development works. Since the course is aimed at a continuous learning audience, no prior programming experience is required and the basics of programming are taught when discussing JavaScript.

The participants came from very diverse backgrounds, from high school without prior experience in programming to well established professionals in the industry who were looking to expand their knowledge within the web domain. Overall, there were 127 students who completed the enrolment survey with 78 students passing the course, giving the passing rate of 64%. Interestingly, over half of the students identified as female in the enrolment survey[1].

4.2 Material Development

The course was built for A+ platform. The main content was developed with ReStructuredText with some custom directives built within the department. A+ itself does not grade exercises but rather sends them to external services that focus on grading submissions for particular exercise types. Most of our exercises were realized either using mooc-grader[2] or Acos [21] platforms. In addition, Acos hosted various other types of online learning activities, such as conceptual visualizations related to client-server messaging. These used the A+ protocol [16] for communication.

We also used external services for other course functionalities through the LTI protocol [7]. These services were the question & answer platform where teaching assistants provided help during set times, and a code submission tool that enabled students to submit their code to teaching assistants without revealing it to other students or the internet at large.

[1] We suspect that the particular context for introductory programming had something to do with this. We also had a number of students describing their background being in UX and web design related jobs. A more comprehensive study of the course is planned.
[2] https://github.com/apluslms/mooc-grader

4.3 Issues in COLM Development

During development of this new online material we recorded multiple issues in our weekly discussions. After the course was completed we carried out a case study of development team experiences and identified the following issues stemming from the complexity of the environment and material:

- *Lack of documentation* led to a frustrating developer experience and wasted time. Some of the components and subsystems were written to accomplish a particular task but the documentation for those were missing or lacking in some way. We were also guilty of this ourselves, when we developed new exercise types in a hurry and did not document them. We can use these systems but sharing them with other instructors is problematic.
- We had automated graders for student submissions but testing their functionality relied completely on the instructors *manually testing* variations of solutions and checking that the received points matched. Inevitably, this led to bugs in the production where students either did not receive full points for correct solutions or were awarded too many points for (partially) incorrect submission.
- The overall development experience suffered greatly from long *compile times*. For any change, the whole material needed to be compiled, which took tens of seconds to do. There is no hot reloading support in the system, so after the compilation the browser would need to be manually refreshed to see the results. This is particularly cumbersome when making small changes.
- The use of multiple *programming languages* also complicated development, and in particular setting up tooling for development. In our case, the material was written with ReStructuredText, with additional directives written in Python, and naturally we had some JavaScript to handle tasks in the browser. In addition to this, we had exercise graders written with Python as well as Node.js based JavaScript.
- While most of the content and exercises resided in the main course repository, we still had *multiple repositories* to host specific types of smaller exercises. While the overall material can be updated by pushing to a specific branch on the main repository, all the other repositories need to be managed manually.

Though we encountered difficulties during material development and could not achieve everything we aimed for, we were still able to adapt and edit the material to our liking, and we began on a path toward a more refined process for material creation.

The issues we encountered led us to conduct a small scale interview study during spring 2020. We asked instructors from various universities in Finland about their course material development experiences and approaches. The main themes arising from these interviews are discussed in the next section.

5 INTERVIEWS

5.1 Interview Context

As discussed in section 4, issues in COLM development led to the small interview study, which was conducted in spring 2020. The aim of the study was to collect CS instructor experiences from various Finnish universities regarding Learning Management Systems, other online platforms, and material development.

There were altogether seven (7) interviewees from three (3) Finnish universities. The age of interviewees varied from about 30 years to about 50 years. Most of the interviews were oral interviews, and one was conducted in written form. The length of oral interviews differed from 25 minutes to 56 minutes. The interviews were semi-structured and there was a certain set of predefined open-ended questions dealing with the main concepts, use cases, issues and ideas related to material development in CS courses and usage of LMSs. Along with predefined questions, some of the questions were also improvised during the interviews based on the themes which were discussed in the interview process. Generally, the themes and issues discussed in interviews were usually associated with the research topics and interests of the interviewee.

Interviews were conducted via Zoom[3] application and they were recorded and later analyzed thematically. Interviews were not transcribed word by word, but instead we used a thematic analysis to identify and extract themes and issues regarding LMSs and material development in CS courses. We discuss the most essential of them next.

5.2 Themes and Issues

One major theme was **compile times** within the material development process. As material development with ReStructuredText (RST) requires compiling the material each time we want to see the changes applied to the development environment, a duration of this compile process is important. The compile process can feel exhaustingly long considering the amount of changes that require compilation.

For example, a simple typo fix requires compiling all files, which is not effective especially if we consider that it may break a good "flow state" and cause frustration. This issue was recognized by several interviewees. One proposed solution was regarding the markup language used and RST issues: utilizing strong typing during material development, for example by using specific programming languages like TypeScript or Elm, would alleviate this.

Additionally, **decreasing the need for manual** and visual **checking** of the results after the compile process (such as is the material displayed correctly, are links and feedback of multiple-choice questions correct etc.), was stated as a very important matter to consider. If we could decrease the need of manually checking for simple issues within the material development workflow, a long compile process would no longer be a problem.

A key theme from the interviews was **developer friendliness**. Especially in CS courses the material developers have usually some kind of background in software engineering or experience in programming as a profession. Therefore, the learning material development process was usually seen as similar to a software development project. The writing process has a "development experience" and it is similar to programming or developing software. This is contradictory to the non-developer teachers and material developers who do not have a background in CS. They may like more WYSIWYG editors or point-and-click style editors within LMSs, even

[3] https://zoom.us/

though some software development environments may offer almost WYSIWYG like experience.

Additional themes we identified from the interviews were **automated testing** of the course exercises, **utilizing analytics**, **preventing plagiarism**, the ability to **customize the look of the course**, and **running own scripts in learning platform**. Abilities to use common software development process tools such as automated tests and continuous integration systems were identified. Additionally, **possibilities to analyze** course **statistics and data** during the course and after it, **comparing course instances** (how does the course differ in comparison to previous instance of the same course) and importance of **accessibility** were mentioned.

6 DISCUSSION

We have presented Complex Online Learning Material – COLM – as a concept. In essence, it is the elaborate software project that is the basis for a modern online CS course. Typically, the material is produced through a software pipeline, with multiple components providing different course functionalities.

Through a case study of our course development as well as a small interview study we have presented key issues currently present in COLM development. Some of the identified issues, such as compile times, lack of automated testing, and developer friendliness, were present in both studies. From the interviews we expanded critical themes with needing to enhance learning analytics, platform customization, and plagiarism prevention.

In practice, we suspect that lack of time, focus on developing new material quickly, and changing course staff (including course assistants, who might also participate in the development of tools) contribute to the fact that best practices of software development are not followed. Rather, new material and systems are developed on an as-needed basis with an ad-hoc approach to documentation.

In conclusion, we suggest that course materials in CS can be a full software project with all the complexities that software development entails. Instead of succumbing to ad-hoc solutions, insufficient testing, and long compile processes we should act more like professional developers when it comes to complex online learning material development.

REFERENCES

[1] A+ LMS. 2020. A+ LMS The extendable learning management system. https://apluslms.github.io/ Accessed: 2020-06-15.

[2] Kamla Ali Al-Busaidi and Hafedh Al-Shihi. 2012. Key factors to instructors' satisfaction of learning management systems in blended learning. *Journal of Computing in Higher Education* 24, 1 (2012), 18–39. Publisher: Springer.

[3] Ibrahim Almarashdeh. 2016. Sharing instructors experience of learning management system: A technology perspective of user satisfaction in distance learning course. *Computers in Human Behavior* 63 (2016), 249–255. Publisher: Elsevier.

[4] Peter Brusilovsky, Stephen Edwards, Amruth Kumar, Lauri Malmi, Luciana Benotti, Duane Buck, Petri Ihantola, Rikki Prince, Teemu Sirkiä, Sergey Sosnovsky, and others. 2014. Increasing adoption of smart learning content for computer science education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*. 31–57.

[5] Isaac Chuang and Andrew Ho. 2016. HarvardX and MITx: Four years of open online courses–fall 2012-summer 2016. *Available at SSRN 2889436* (2016).

[6] Hamish Coates, Richard James, and Gabrielle Baldwin. 2005. A critical examination of the effects of learning management systems on university teaching and learning. *Tertiary education and management* 11 (2005), 19–36.

[7] IMS Global Learning Consortium. 2010. Learning Tools Interoperability. http://www.imsglobal.org/toolsinteroperability2.cfm. Accessed 2020-10-12.

[8] Christian Dalsgaard. 2006. Social software: E-learning beyond learning management systems. *European Journal of Open, Distance and e-learning* 9, 2 (2006).

[9] John Daniel. 2012. Making sense of MOOCs: Musings in a maze of myth, paradox and possibility. *Journal of interactive Media in education* 2012, 3 (2012). Publisher: Ubiquity Press.

[10] Aracele Garcia de Oliveira Fassbinder, Marcelo Fassbinder, Ellen Francine Barbosa, and George D Magoulas. 2017. Massive open online courses in software engineering education. In *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[11] Ryann K Ellis. 2009. Field guide to learning management systems. *ASTD learning circuits* (2009), 1–8.

[12] Ángel Fidalgo-Blanco, María Luisa Sein-Echaluce, and Francisco José García-Peñalvo. 2016. From massive access to cooperation: lessons learned and proven results of a hybrid xMOOC/cMOOC pedagogical approach to MOOCs. *International Journal of Educational Technology in Higher Education* 13, 1 (2016), 24. Publisher: Springer.

[13] Antonio Fini. 2009. The technological dimension of a massive open online course: The case of the CCK08 course tools. *The International Review of Research in Open and Distributed Learning* 10, 5 (2009).

[14] Francisco J García-Peñalvo, Ángel Fidalgo-Blanco, and María Luisa Sein-Echaluce. 2018. An adaptive hybrid MOOC model: Disrupting the MOOC concept in higher education. *Telematics and Informatics* 35, 4 (2018), 1018–1030.

[15] Phil Hill. 2017. Academic LMS Market Share: A view across four global regions. https://eliterate.us/academic-lms-market-share-view-across-four-global-regions Accessed: Feb 2020.

[16] Ville Karavirta, Petri Ihantola, and Teemu Koskinen. 2013. Service-oriented approach to improve interoperability of e-learning systems. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*. IEEE, 341–345.

[17] Alexander McAuley, Bonnie Stewart, George Siemens, and Dave Cormier. 2010. The MOOC model for digital practice. *Social Sciences and Humanities Research Council's "Knowledge Synthesis Grants on the Digital Economy* (2010).

[18] Tanya J McGill and Jane E Klobas. 2009. A task–technology fit view of learning management system impact. *Computers & Education* 52, 2 (2009), 496–508.

[19] Rolin Moe. 2015. The brief & expansive history (and future) of the MOOC: Why two divergent models share the same name. *Current issues in emerging elearning* 2, 1 (2015), 2.

[20] Martin Pärtel, Matti Luukkainen, Arto Vihavainen, and Thomas Vikberg. 2013. Test my code. *International Journal of Technology Enhanced Learning 2* 5, 3-4 (2013), 271–283.

[21] Teemu Sirkiä and Lassi Haaranen. 2017. Improving online learning activity interoperability with acos server. *Software: Practice and Experience* 47, 11 (2017), 1657–1676.

[22] Thomas Staubitz, Hauke Klement, Jan Renz, Ralf Teusner, and Christoph Meinel. 2015. Towards practical programming exercises and automated assessment in Massive Open Online Courses. In *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE, 23–30.

[23] The Finnish Institute of Technology (FITech). 2020. Studies. https://fitech.io/en/studies/ Accessed: 2020-10-12.