Designing a unique revision loop updating courses simultaneously on different MOOC platforms

Ella Hamonic, Rémi Sharrock, Petra Bonfert-Taylor, Michael Goudzwaard, Gérard Memmi, Catherine Chow, Josh Meise

Designing a unique revision loop updating courses simultaneously on different MOOC platforms

Ella Hamonic
Telecom Paris & Le Monde Après
Paris, France
0000-0001-6158-1572

Rémi Sharrock
Telecom Paris
Paris, France
0000-0001-8952-8933

Petra Bonfert-Taylor
Thayer School of Engineering
Dartmouth College
Hanover, NH, USA
petra.bonfert-taylor@dartmouth.edu

Michael Goudzwaard
Dartmouth Center for the Advancment of Learning
Dartmouth College
Hanoer, NH, USA
0000-0002-1240-7348

Gérard Memmi
Telecom Paris
Paris, France
0000-0002-3380-8394

Catherine Chow
Dartmouth College
Hanover, NH, USA
catherine.m.chow.24@dartmouth.edu

Josh Meise
Dartmouth College
Hanover, NH, USA
joshua.m.meise.24@dartmouth.edu

***Abstract*— This study introduces a content revision loop for simultaneous course updates across MOOC platforms. It uses a single source of truth and iterative, data-driven methodologies, enabling efficient dissemination of updates and assessment improvement. The approach provides a model for instructors managing courses on multiple platforms.**

Keywords— LTI, single source of truth, feedback collection process, revision, design, learning programming

I. INTRODUCTION

The dissemination of educational courses across numerous MOOC (massive open online course) platforms can be simplified through the incorporation of the LTI (Learning Tools Interoperability) protocol. Implementing a centralized content hosting system also facilitates simultaneous course content updates and continuous enhancement of assessments. The present study scrutinizes the development of a content revision pipeline, which employs iterative and data-driven methodologies to optimize course updates.

In this paper, we will delve into the course architecture, examining the intricate dynamics between learners, the course team, and developers as they collaboratively interact and exchange information to enhance the course quality. The uniqueness of the course architecture lies in its ability to offer the computer programming MOOC course series simultaneously on edX, Coursera, and on-campus LMS through the France-IOI learning platform and interactive tools to learn programming [1].

Our analysis focuses on how this architecture supports feedback collection, iterative revisions, and continuous improvements. Two key research questions are:

- How can we efficiently coordinate a diverse team of teaching assistants across different platforms and institutions for efficient forum monitoring and issue reporting?
- How does a single source of truth for course content enhance quality, consistency, and adaptability of materials, and what are the associated benefits and challenges of multi-platform implementation?

These questions will help optimize the course improvement efforts, ensuring a high-quality learning experience across platforms.

II. OFFERING COURSES ON MULTIPLE PLATFORMS USING A SINGLE SOURCE OF TRUTH WITH A CENTRALIZED CONTENT MODIFICATION

A. Multiple Sources Versus Single

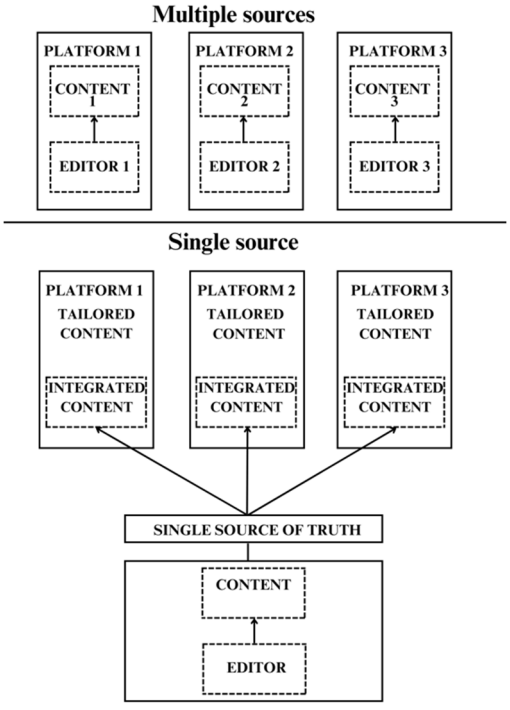

Fig. 1 Multiple Sources vs Single Source

We evaluated the pros and cons of using a single source of truth (via LTI or iframe integration) for updating course content across multiple MOOC platforms, such as Coursera, edX, Moodle, or Canvas, versus using built-in editing tools within each platform. As depicted in Figure 1, built-in tools often lead to content duplication and risk of inconsistencies, while a single source provides centralized, consistent content across platforms. It also allows for seamless course migration across platforms, unlike built-in editors, which create strong platform dependency.

With a single source, course updates are simplified as changes made to the central content are reflected across all platforms. This method enables centralized modifications to course material, assessments, instructions, and documentation, reducing complexity and streamlining the editing process. Conversely, built-in tools require separate updates for each content instance, making the process time-consuming and error-prone.

The use of a single source ensures content stability amid constant MOOC platform evolution, offering a consistent learning experience. Built-in tools may result in unexpected disruptions due to changes in platform functionality. Moreover, a single source allows simultaneous presence on multiple platforms, maximizing course exposure. However, platform-specific requirements must be manually configured, and some limitations, such as customization of grading criteria, exist.

In summary, while built-in tools offer advantages, using a single source for updating courses across platforms provides numerous benefits. These include content consistency, reduced platform dependence, simplified update process, streamlined editing, minimized disruptions, cost optimization, and enhanced course exposure.

B. Revision loop for iterative improvements

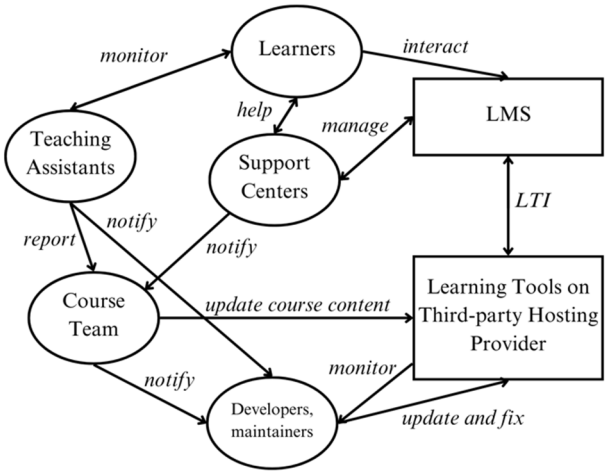

Fig. 2 Revision loop to improve the course quality

Our design choices for the revision loop were inspired by the CI/CD (Continuous Integration and Deployment) practice [2]. In the revision loop for iterative improvements, we will consider dynamicity as an important characteristic when designing a content update architecture. We also consider that the course team works co-jointly with some course content developers or tool developers that manage the learning tools hosted on third party providers. Continuous monitoring of the course and prompt issue reporting play a crucial role in ensuring its smooth operation. As shown in Figure 2, we suggest that in the context of MOOC and to enable scalability, some Teaching Assistants (TAs) actively monitor the learners via forums across various MOOC platforms and employ multiple channels to report issues to the course team. The best practice would be to have at least one Teaching Assistant dedicated to one platform. Urgent matters, such as tool breakdowns, must immediately be communicated via email or a phone call to the course team, ensuring timely attention and resolution. Additionally, regular meetings provide a dedicated space for TAs to provide updates on ongoing issues and discuss potential solutions and must be scheduled in the long term. A ticketing system, such as GitHub Issues, is a good practice to track and manage reported issues systematically, allowing for efficient collaboration within the course team. An example of using GitHub project management features is given in [3] for the introductory computer programming course CS50 at Harvard University.

To gather further insights, the course team can leverage the different MOOC platform's support centers, which provides valuable information if students have reached out with specific concerns or inquiries. Moreover, platforms like Coursera often feature other types of feedback systems tailored to each component of the course, enabling learners to provide feedback directly. This feedback system serves as an additional source of information for the course team to identify areas of improvement or address any issues reported by students. When urgent bugs or critical issues are identified, the course team can notify the developers or third party provider promptly, triggering necessary fixes or updates to the tools or platform infrastructure.

In addition to human monitoring, developers and maintainers of the third party tools have access to server metrics, allowing them to closely monitor the hosting environment as it is crucial to have a scalable infrastructure for the third-party tools. This access enables them to proactively identify any potential server-related issues or anomalies and promptly notify the course team. By staying informed about server performance and addressing potential infrastructure concerns, the course team can maintain the stability and availability of the course tools.

In summary, continuous monitoring is facilitated by Teaching Assistants who monitor forums, report issues via email, weekly meetings, and a ticketing system. The course team also benefits from insights gathered through the MOOC platform's support center, feedback systems, and server metrics. By employing a comprehensive monitoring approach, the course team can promptly address issues, enhance the learning experience, and ensure the course operates smoothly at scale and throughout its duration.

C. Single Source Of Truth, Versioning and Backup

Content editing, tool updates and any other modification of the content or the third-party tools takes place based on learner feedback and ongoing improvements identified by the course team. We suggest having within the process a versioned, backed-up single source of truth, which serves as the authoritative repository of the course content. Versioning allows

for easy access to previous versions, enabling the team to revert to an older version if necessary or understand who and why a modification of a content occurred in the past. Maintaining a comprehensive history of all content changes and having access to a global historical record is invaluable for the course team as it provides visibility into the evolution of the course materials over time, enabling them to track modifications, view the delta of changes, and understand the progression of content development. The ability to trace the history of changes not only ensures transparency and accountability within the team but also facilitates effective collaboration and the ability to analyze the impact of modifications on the overall course structure and content quality.

Furthermore, a backup system must be in place to ensure the integrity and availability of the course content, safeguarding in the long term the learning experience for students and minimizing any potential downtime.

D. Extending the Framework for Revision of MOOC Assessments

We also build our approach upon the Framework for Revision of MOOC assessments created by C. M. Friend, M. Avello, M. E. Wiltrout and D. G. Gordon [4]. The framework provides a structured approach to revising formative assessments in MOOCs. The framework aims to guide instructors in evaluating and revising their online course assessments using data-driven approaches. The framework involves analyzing learner performance data, evaluating the quality of formative assessments, identifying common learner errors, and revising assessments based on the findings. This iterative, data-driven approach aims to improve the effectiveness of formative assessments in subsequent course runs. The paper outlines the steps of the framework for revising MOOC formative assessments.

These steps shown in Figure 3 include identifying low-performance questions, evaluating the causes of low performance (instructor-based, learner-based, or both), revising the questions based on the identified causes, and iterating the process in subsequent course runs. The authors provide specific strategies for revising low-performance questions, such as adjusting coding, modifying wording, decomposing complex questions, emphasizing key areas in instructions, and providing context-specific hints.

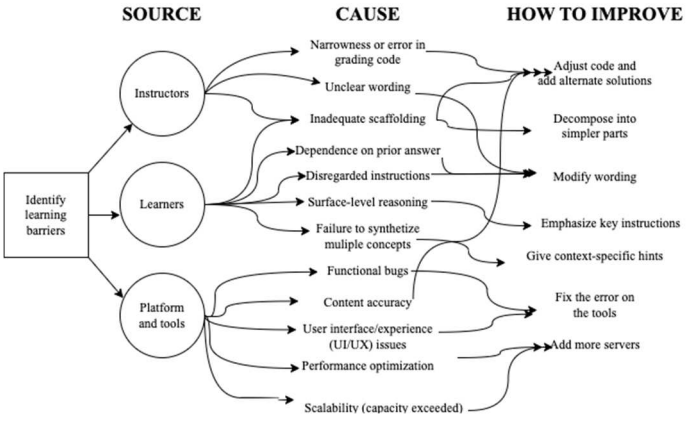

Fig. 3 Extended framework of formative assessment revision

This paper extends the framework in Figure 3 by incorporating additional sources of learning barriers. The extended framework aims to proactively identify educational barriers by considering various metrics that assess the effectiveness of online learning experiences.

In addition to instructor- and learner-based causes, the framework also considers the source of barriers, which can originate from platforms or tools used in the online courses. The causes of low performance may include functional bugs, content accuracy issues, user interface/experience problems, performance optimization challenges, compatibility issues, and scalability limitations such as capacity exceeding.

To improve the online course, the framework suggests applying the revision loop, which involves addressing errors on the platform or tool, making necessary fixes, enhancing content accuracy, improving user interface/experience, optimizing performance, and resolving compatibility and scalability issues.

The process of improvement may include actions such as fixing bugs in the online learning platform, updating and refining content, enhancing the user interface to provide a better experience, optimizing performance by implementing technical improvements, and addressing compatibility issues across different devices and browsers.

Scaling up the infrastructure by adding more servers or expanding the capacity can also be a part of the improvement process to ensure the online course can accommodate a larger number of learners without compromising performance.

III. Applying the Architecture to a Computer Science Course Series on Coursera, edX and Local LMS

In this section, we will explain the steps we took to implement the previously described architecture in order to deploy a series of courses on multiple MOOC platforms. Additionally, we will demonstrate how we incorporated a revision loop to consistently enhance the material across these platforms. Therefore, we will first introduce the course series in subsection A, followed by the platform in subsection B, and finally, we focus more on the revision loop in subsection C.

A. The C Programming with Linux Course Series

The "C Programming with Linux" course series, a joint effort between Dartmouth College and Institut Mines-Télécom (IMT in France), was created in 2017. This comprehensive program aims to provide individuals with a strong foundation in C programming and the Linux operating system. The series consists of seven courses endorsed by prominent global high-tech companies, including Nokia Bell Labs, Airbus, Thales, and Gainwell Technologies. Designed to facilitate the onboarding of beginners, these courses introduce the fundamentals of C programming, even for those with no prior programming experience. Learners gain hands-on experience writing, reading, and debugging computer programs in C, while also familiarizing themselves with the Linux operating system.

The course series offers a unique feature that allows beginners to immediately start coding in C through web browser-based coding tools, eliminating the need for

installation. The significance of learning C programming lies in its widespread usage in various technologies, including smartphones, navigation systems, robots, drones, trains, and electronic devices. C is the preferred language in scenarios where speed and flexibility are critical, such as embedded systems and high-performance computing. As one of the foundational programming languages, it continues to be taught in engineering schools worldwide and remains a stable and popular choice among programmers.

Additionally, the course series covers the Linux operating system, which is widely used by computer scientists and developers. Linux powers supercomputers, servers, Android devices, and most internet of things (IoT) devices. The course series provides an introduction to the Linux command line and essential Linux tools for C programmers. These skills are highly sought after in today's tech industries. The program offers guided exercises, coding demonstrations, and more elaborate assignments to provide learners with practical experience and proficiency in C programming.

The "C Programming with Linux" course series initially launched as a Professional Certificate on edX in 2018, won the edX prize in 2019, offering learners the opportunity to earn a recognized certification upon completion. Building upon its success, the series expanded its reach and was introduced as a Coursera specialization in 2022. This enabled a wider audience to access the course content. Additionally, the course series has been adopted and utilized on local Learning Management Systems (LMS) within each partnering institution, ensuring accessibility and flexibility for learners associated with Dartmouth College and Institut Mines-Télécom (IMT). Whether through edX, Coursera, or local LMS platforms, the course series has been made available to learners across various educational environments, catering to their specific needs and preferences.

Because the team had previously developed a French version of the content on the FUN-MOOC platform (the French government public instance of edX) we already had experience developing and integrating LTI content into an open-edX platform. When the team decided to deploy the course series on Coursera a few years later, they worked closely with the Coursera developers to try a new beta feature to automatically import content from an edX course export. Because of the difficulty encountered during this import/export process, we decided to focus on using a single source of truth and came up with the “single source” architecture presented in section II, Figure 1.

B. The implementation of the single source of truth

The implementation of the single source of truth architecture involved close collaboration between the course team and developers associated with France-IOI, a non-profit organization dedicated to supporting educational initiatives and developing open-source tools for programming [5]. The course series emphasizes problem-solving through programming exercises, with online formative assessments playing a critical role in facilitating engagement and learning in MOOCs [4]. To support this approach, the team developed LTI specific tools for the purpose and integrated them using the LTI modules for each platform.

Taskgrader, an open-source autograding tool [6], provides instant feedback in large-scale online programming classes, offering extensive feedback to student code submissions. It is also capable of creating quizzes and we only used this tool to create quizzes across all platforms. Codecast [1], an in-browser C language interpreter, is paired with an event and voice recorder and player, synchronizing audio with source code edition, visualization, step-by-step execution, and testing, facilitating teaching and learning programming concepts. Weblinux [7], a web app tool, offers a client-side and offline Linux OS environment within the browser, allowing learners to experiment with Linux without the need for software installation.

To maintain a single source of truth and enable continuous improvements, the team developed a centralized editing interface. This interface allows for updates to Taskgraders, Codecasts, and Weblinux while maintaining a versioned, single source of truth within a GIT repository. The repository is backed up on Amazon S3 for secure storage. By using this architecture, the course team avoids the proliferation of multiple copies of content with different formats, ensuring consistency across platforms.

For a detailed visual representation of the specific C programming MOOC series architecture, refer to Figure 4.

C. The implementation of the revision loop

Within our specific context of offering courses on C programming with Linux through multiple platforms (Coursera, edX, Moodle, and various local Learning Management Systems), the revision loop operates within a coordinated and international ecosystem. Learners, including registered individuals from Coursera, edX, and our respective local universities (Dartmouth College in the USA and IMT in France), engage with the course content across these platforms. They interact through platform-specific built-in forums or directly with local professors, asking questions, providing feedback, and seeking support as needed.

To ensure effective monitoring of these forums, we have a team of over 15 teaching assistants, with a maximum of four working simultaneously, who maintain a coordinated schedule. The course team itself consists of professors and learning designers from the two institutions, bringing together international expertise from the United States and France. Weekly meetings are conducted to address issues, track progress, and implement necessary actions continuously since 2018.

Involving undergraduate students in the course series has been a priority for the course team. To date, 14 undergraduate students have been trained to work with the MOOCs, gaining valuable experience in course design, content creation, and online teaching [8]. These students have actively contributed to the revision process and played a vital role in ensuring that the courses meet the diverse needs of the audience. Additionally, the course team has collaborated with a community teaching assistant (TA) to enhance the learning experience for learners within the community. By involving undergraduate students and community members in the course revision process, the course team has not only expanded access to high-quality

education but also fostered a culture of collaboration and knowledge-sharing.

To facilitate feedback collection and organization, the course team set up a GitHub repository, leveraging the platform's issue tracking system. The team has implemented a robust feedback collection system by utilizing a GitHub repository. This feedback from learners and teaching assistants is then used to inform updates and revisions to the course content, allowing the instructor team to identify and address bugs, provide better hints and possible solutions to assessments, and overall enhance the quality of the course. The course team recognized the importance of collecting feedback from learners and teaching assistants to continuously improve the course content. A systematic process was established to ensure the quality of the learning experience and address any issues or challenges faced by the learners.

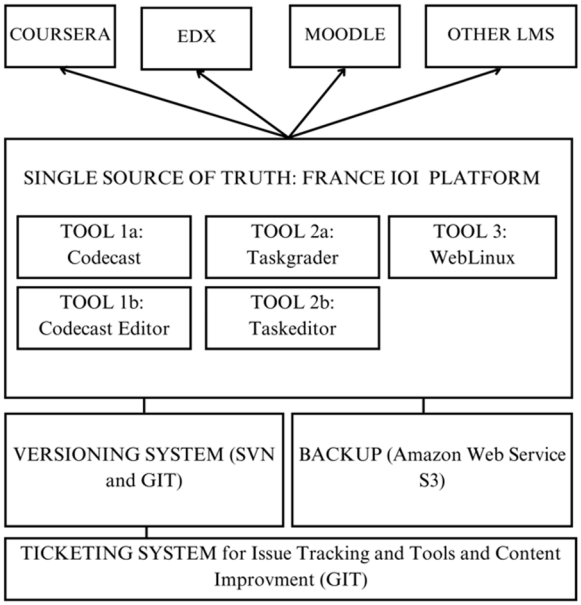

Fig. 4 C Programming with Linux architecture

Teaching assistants were encouraged to report any issues, provide suggestions, or ask questions through the GitHub repository. The team established guidelines for submitting issues and feedback, ensuring that it was comprehensive, clear, and actionable: what exactly are we talking about, how to access it, how to reproduce a bug, when an event happened.

According to Hooimeijer and Weimer [9], bug triage is the act of inspecting an issue or bug report, understanding its contents, and making the initial decision regarding how to address the report. To ensure efficient bug triage and effective course improvement, the course team has established a systematic process for handling reported issues using specific tags and categories and to ensure that no reported issues are overlooked or duplicated.

The bug triage process involves assigning priorities to each bug based on its impact and urgency. Urgent issues that severely hinder the learning experience or functionality of the course are given immediate attention and allocated the necessary resources for prompt resolution. These could include critical tool breakdowns, major content inaccuracies, or critical UI/UX issues.

For bugs that are categorized as long-term improvements, the course team conducts a thorough analysis to identify patterns and recurring issues. They carefully review learner feedback, conduct surveys, and monitor user engagement metrics to gain insights into areas that require significant enhancements. By considering the feedback and prioritizing long-term improvements, the team can allocate resources effectively and focus on addressing the most impactful issues.

Bug triage techniques offer valuable support to the course team in their endeavor to continuously enhance the course. By implementing effective bug triage, the team can prioritize issues based on their urgency and impact. Urgent matters requiring immediate attention, such as critical tool breakdowns or severe grading problems, can be promptly addressed to minimize disruption to learners. Similarly, short-term, low-effort actions, such as fixing typographical errors or addressing minor accessibility issues, can be quickly resolved to ensure content quality.

During the bug triage process, the course team also collaborates with technical experts or developers to address platform-specific problems or technical issues. This collaboration ensures that compatibility issues across different platforms, browsers, or devices are identified and resolved. The team thoroughly tests the course on various configurations and environments to ensure a consistent and seamless learning experience for all learners.

Long-term improvements can also be identified through bug triage. By analyzing recurring issues or feedback, the course team can identify patterns and prioritize long-term enhancements that have a substantial impact on the course experience. For example, if learners consistently struggle with a particular concept, the team can allocate resources to develop additional explanatory materials or supplementary resources to address this challenge comprehensively. Furthermore, bug triage can also aid in identifying and prioritizing platform-specific problems, technical issues, or suggestions for improving the overall user experience, thereby guiding the team's efforts in creating a more seamless and engaging learning environment.

GitHub issues were used to collect and manage feedback effectively. Each issue represented a specific problem, suggestion, or question raised by the learners or teaching assistants. The team categorized the issues based on their nature, such as bug reports, feature requests, or general feedback. This categorization helped in prioritizing the issues and addressing them efficiently.

Bug triage in the context of course improvement can encompass various aspects, such as:

- Functional Bugs: These are issues that affect the core functionality of the course, such as broken links, non-functioning interactive elements, or dysfunctional features within the learning platform. Categorizing these bugs helps prioritize fixes that directly impact the learners' ability to navigate and interact with the course materials effectively.

- Content Accuracy: Bugs related to content accuracy involve incorrect information or inaccuracies in course materials. Categorizing such bugs helps ensure that the content aligns with the latest knowledge, facts, and industry standards, ensuring a high-quality learning experience for the students.
- User Interface/Experience (UI/UX) Issues: Bugs in this category pertain to problems with the course's visual design, layout, or user interface. Examples include inconsistent formatting, confusing navigation, or accessibility issues. By categorizing UI/UX issues, the course team can prioritize improvements that enhance usability, readability, and overall user experience.
- Performance Optimization: These bugs refer to issues that impact the course's performance, such as slow loading times, high resource utilization, or inefficient use of network bandwidth. Prioritizing these bugs allows the team to optimize the course's technical aspects, ensuring smooth and responsive performance for learners across different devices and network conditions.
- Compatibility Issues: Bugs related to compatibility encompass problems that arise when the course is accessed on different platforms, browsers, or devices. This could involve issues with rendering, functionality, or responsiveness on specific configurations. By categorizing compatibility issues, the course team can prioritize efforts to ensure the course functions consistently and seamlessly across various platforms and environments.
- Bug triage also extends to platform-specific issues, such as financial aid. Identifying and addressing bugs related to financial aid can help ensure that eligible learners can access the necessary support to participate in the course effectively.

By incorporating bug triage techniques and addressing these various aspects, the course team can continuously improve the course, enhance the learning experience for students, and create a more robust and reliable educational offering, even in the long term. 782 issues were created since October 2018.

VI. Conclusion: Continuous Improvement through Data-driven Design

The course team demonstrated a strong recognition of the value of data in informing the revisioning process and improving the learning experience [8]. By systematically collecting and analyzing learner feedback and assessment data from various sources, including platforms such as Coursera, edX, Moodle, and local Learning Management Systems, the team gained valuable insights into the effectiveness of the course content. This learner-centered approach supported evidence-based decision-making and facilitated continuous improvement of the course material.

The utilization of a single source of truth architecture proved to be extremely beneficial in our specific use case, despite the heavy development involved [5]. This approach ensured that improvements made to the course content automatically filtered into all offerings of the course, regardless of the particular platform. The single source of truth reduced the complexity of managing multiple copies of content and streamlined the revision process, allowing for efficient updates and enhancements.

Moving forward, further investigation is needed to understand the impact of utilizing multiple MOOC platforms (Coursera, edX, Moodle) along with local Learning Management Systems on student learning experience and engagement. Additionally, exploring the contribution of teaching assistants (TAs) and their impact on course quality [10], as well as striking the right balance between using third-party content creation tools and MOOC platforms built-in tools, are areas that warrant continued examination.

Through the implementation of a single source of truth architecture and the utilization of learner feedback and assessment data, the course team has demonstrated a commitment to providing an optimal learning experience for students. By continuously improving the course content and leveraging data-driven insights, we can enhance the educational journey for learners and ensure the course remains effective and engaging.

References

[1] R. Sharrock, E. Hamonic, M. Hiron, & Carlier, S, 2017,. Codecast: An innovative technology to facilitate teaching and learning computer programming in a C language online course. In *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale* (pp. 147-148).

[2] M. Shahin, M. A. Babar, & L. Zhu, L. 2017 Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. IEEE access, 5, 3909-3943.

[3] D.Malan, J. Sharp, C., van Assema, B. Yu, & K. Zidane. (2021, March). CS50's GitHub-Based Tools for Teaching and Learning. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 1354-1354).

[4] C. M. Friend, M. Avello, M. E. Wiltrout and D. G. Gordon, "A Framework for Revision of MOOC Formative Assessments," 2022 *IEEE Learning with MOOCS (LWMOOCS)*, Antigua Guatemala, Guatemala, 2022, pp. 151-154, doi: 10.1109/LWMOOCS53067.2022.9927973.

[5] R. Sharrock, E. Collin,T. Labat, E. Hamonic, P. Bonfert-Taylor, M. Goudzwaard, Teaching and Learning Programming with Linux using In-Browser Client-Side Web Technologies: Exploring the Key Features for Achieving Systems and Tools Scalability, *L@S '22: Proceedings of the Ninth ACM Conference on Learning @ Scale* June 2022 Pages 427–430 doi.org: 10.1145/3491140.3528295

[6] R.Sharrock, P, Bonfert-Taylor, M. Hiron, M. Blockelet, C. Miller, C, M. Goudzwaard, E, Hamonic, 2019, June, Teaching c programming interactively at scale using taskgrader: An open-source autograder tool. In *Proceedings of the Sixth (2019) ACM Conference on Learning@ Scale (pp. 1-2)*.

[7] Sharrock R., Angrave L., & Hamonic E.. 2018. WebLinux: a scalable in-browser and client-side Linux and IDE. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale (L@S '18)*. Association for Computing Machinery, New York, NY, USA, Article 45, 1–2. https://doi.org/10.1145/3231644.3231703

[8] S. Labrique, D. Ducarme, B. Raucent, B., Se former en ligne au Tutorat : un défi pour les assistants-chercheurs, 2021, Les Annales De QPES, 1(4). https://doi.org/10.14428/qpes.v1i4.62423

[9] Hooimeijer P. & Weimer W.. 2007. Modeling bug report quality. In *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering* (ASE '07). Association for Computing Machinery, New York, NY, USA, 34–43. https://doi.org/10.1145/1321631.1321639

[10] T. Bouvy, M. de Theux, B. Raucent et al., « Chapitre 14. Compétences et rôles du tuteur en pédagogies actives », dans : Benoît Raucent éd., Accompagner des étudiants. Quels rôles pour l'enseignant ? Quels dispositifs ? Quelles mises en œuvre ? Louvain-la-Neuve, De Boeck Supérieur, « Pédagogies en développement », 2010, p. 371-396. DOI : 10.3917/dbu.rauce.2010.01.0371. URL : https://www.cairn.info/accompagner-des-etudiants--9782804133313-page-371.htm