Courseware as Code

Setting a new bar for transparency and collaboration

Abstract— **This Innovative Practice Category Work In Progress proposes creating and managing all courseware as code. Based on our school's success with this approach in achieving rapid synchronized collaboration, we recommend educational organizations consider adopting this idea. While DevOps and Continuous Integration/Continuous Deployment (CI/CD) have brought software and network engineering into the twenty-first century, academic courseware has not seen organization-wide process improvement at the same scale. Inspired by industry best practices, our school chose to develop courseware flexibly within a GitLab instance. Through GitLab, we facilitate change discussions, provide transparency in updates, and allow faculty, students, and even the workforce to identify issues and suggest content. This enables space for necessary, creative, and even innovative changes by giving all users a voice in the change process and setting a culture of contribution.**

In its first year, Courseware as Code enabled over 250 students in 30 classes to benefit from standardized yet flexible and up-to-date content. For our core course, 29 instructors over up to three geographically disparate locations stayed engaged with 5,085 commits and 113 issues raised and discussed. To our knowledge, no other similarly sized and divided group of instructors has achieved this level of collaboration towards synchronized courseware.

Index Terms—**courseware, version control, educational technology, collaborative software, distributed management**

I. Introduction

In 2014, we were given a task to stand up a school to meet a workforce need. As we began to create content as quickly as possible (often described as 'building an airplane while in flight' by a senior school official), we turned to cloud-hosted storage and synchronization. With many contributors but without rigorous control on groups, permissions, and content visibility and acceptance, we recognized our need for increased structure and improved workflows. Through a GitLab instance, we chose to implement an 'Everything as Code' mentality to both our virtual training resources and labs and our courseware.

Reviewing existing documented uses of distributed revision control systems (DVCS) did not reveal a courseware synchronization use case that matched our own. In 2007, the Rose-Hulman Institute of Technology used a centralized version control system (VCS) to synchronize two CS1 classes between two instructors and one teaching assistant [1]. They noted that the VCS simplified sharing materials, reduced duplicative content development, and provided transparency to contributions. While they recommended VCS for widespread adoption, they did not publish additional work on iterative improvements or teaching base expansions. In 2013, an Arizona State University (ASU) group created a web interface to simplify instructor interaction with a DVCS through a set workflow but they did not publish results on the efficacy of the interface [4]. Continuing in 2017, ASU proposed a new web framework with a GitHub backend in which instructors use markdown through SimpleMDE [5]. As before, they did not use their proposed system in a class, settling instead for a faculty review of the framework's potential. Other published cases focused on teaching students to use VCS although many educators also noted the benefits to courseware improvement. In 2013, a Massachusetts-based group implemented git in a CS1 class that included non-CS engineering students [3]. They found even the non-CS majors settled quickly into using DVCS, never questioned the value of learning DVCS, and cited its benefits and relevance to job skills. In 2014, a professor in Ireland used git for second and third year CS students based on industry demand for job applicants with VCS experience [2]. His students surprisingly preferred the Bash git command shell over GUI tools or Visual Studio integration because it allowed them to engage at the level of their ability and increase their understanding and skill without limitations. A 2015 research paper on the use of GitHub in education found that DVCS surpassed traditional learning management systems (LMSs) by enabling additional interactions to include students contributing to course materials instead of only viewing them [6]. The authors of that research identified limitations in DVCS adoption due to a lack of support for PDFs and LaTeX and no existing measurements of student contribution or the acutal efficacy of the sense of a participatory culture. With our implementation of GitLab and its automated workflows as the courseware repository for our multi-instructor, multi-contributor, location-disparate classes, we believe we have solved many of the limitations to using DVCS for education.

II. Courseware as Code concept

Using a DevOps and Continuous Integration/Continuous Deployment (CI/CD) philosophy, our school chose a GitLab instance to maintain both our configuration management for infrastructure and our curriculum artifacts for five primary courses: our four programming modules (C, PowerShell, Bash, and Python) and our technical core course. Our core course engages students in problem solving, research, and gaining a situational understanding of cyberspace security through the topics of Windows, Linux, and networking. For each of these courses, we established a structure of public, internal, and private[1] project repositories to allow fine-grained control

[1]Using GitLab visibility levels as project names; see https://docs.gitlab.com/ee/public_access/public_access.html

of user groups to encourage collaboration while maintaining accreditation-required controls and material integrity.

By leveraging applicable aspects of the software development lifecycle for curriculum management, we envisioned facilitating discussion of suggested changes amongst faculty, transparency for updates, creating custom workflows based on the complexity of course content, and allowing faculty, students, and members of our workforce to identify issues and contribute content. We wanted to enable substantive changes to occur smoothly by requiring contributors to enter all course content in a machine-readable markup language. To date we have leveraged Asciidoctor; markdown, LaTeX, reStructuredText, or any other document formatting and preparation method would also work effectively. Our concept includes replacing non-markup language file formats (e.g. WYSIWYG slides, word processor documents, spreadsheets) with markup language formats and leveraging CI pipelines to create PDFs, html5 slide decks, and more for distribution.

Using a DVCS would also allow us to create trackable versions of courseware with branches, tags, and other git features to enable quality assessments correlated to time and content changes. The version control process could support zeroing in on specific commit sets to analyze whether a specific change improved or degraded student performance. Applying this idea directly to assessments could allow us to validate test material on a per-question basis and automate test assembly with questions chosen randomly from an objectives-based associative array.

III. Courseware as Code implementation

Although we have not yet realized 100% markup language uniformity or automated analysis of assessment material, we have successfully implemented the primary components of Courseware as Code. In addition to enforcing the standardized project repository structure for our five courses, we focused our implementation on user accounts and permissions, content creation and improvement processes, content delivery, automating workflows, and instructor buy-in. Since we launched it in January of 2017, our implementation has gained momentum and success.

Implementing the Courseware as Code concept structure requires user accounts to which we assign roles and permissions. We initially provisioned accounts manually, but we outpaced our ability to sustain this as our school size grew and workforce members gained interest in participating. To enable our system to support students and the workforce, we needed an automated way to provision user accounts. We designed an account creation and approval process with priority to scalability and simplicity. We leverage an existing and well-maintained workforce-affiliated PKI infrastructure to authorize valid smart cards with current root certificate authority. This automatic process enables users to register and log in to our GitLab instance at the guest user permission level[2] for public projects without requiring individual verification or ticket servicing by IT support staff.

Our project repository structure works with user account permissions to provide controls for content access and editing. Public projects contain freely-available materials that anyone can view with or without authentication. We use the Apache 2.0 license for all of our public courseware to encourage sharing. Internal projects contain material to assist instructors including instructor guides and example solutions but do not contain testing material. During on-boarding, we give our instructors developer-level access to our public and internal projects. Private projects contain official testing and assessment materials. A controlled list of personnel have access to these projects so we may maintain test material integrity and prevent instructors from 'teaching to the test'.

All users authorized for an account in our GitLab instance may request access to contribute to courseware from course managers and senior instructors. If requesting access to an internal or private project, we individually consider the nature of the user's request and require the user to sign both a contributor license agreement and a non-disclosure agreement. Course managers may grant users developer-level access so they may contribute new content and suggested adjustments on branches. Course instructors can review and discuss the merge requests for the branches on the basis of quality, accuracy, and relevancy prior to the course manager or senior instructor approving or denying the requests. Through this process of creating, proposing, discussing and implementing changes in real time, we realize the benefits of CI/CD in developing accurate, relevant, and current courseware.

Although we chose to use GitLab to host our courseware, we also work with our university's LMS, Blackboard, for student content access and assignment submissions. Prior to integrating with GitLab content, the courseware entered into Blackboard lacked version control and historical information on changes. The lack of information caused frustration in instructors not knowing whom to talk to about content concerns and resulted in churn of limited resources. With Courseware as Code, we completely changed that process and improved courseware creativity and efficiency. Although we have not yet achieved continuous delivery of content to Blackboard, our existing GitLab CI pipelines automatically render PDFs. We provide a permanent link to the current version of an activity within the Blackboard framework; students will click on the link and access always updated content. Our current process does require consistent directory structures and filenames; if these change, the instructor must update the link in Blackboard. The permanent links only work for public content, as internal or private content requires GitLab authentication through a privileged account. In the future, we plan to evaluate options for automatic user authentication from the Blackboard LMS to our GitLab instance. Planned versions of the university Blackboard LMS include a REST API through which we could design a CI pipeline to automatically create and load tests at a set time as well as support automating analytic programs for assessments.

[2]GitLab user permission levels available at https://docs.gitlab.com/ee/user/permissions.html

Our current CI implementation uses the GitLab shell-runner that executes individual pipelines inside of a monolithic build environment. This requires IT support staff to pre-install all required packages (e.g. Python, Ruby, TeX Live, Asciidoctor). Users who create new pipelines for new projects may request additional package installation by raising an issue within their GitLab project. While this enables straightforward integration, it does not strictly guarantee reproducibility. We plan to implement a container-driven build environment to allow users to specify and approve their own dependencies. Our common CI pipelines include creating PDF documents with asciidoctor-pdf, creating slides with reveal.js, and using python programs to automate calculations and analysis.

For Courseware as Code to be successful, we had to communicate the vision to all of our faculty and make it accessible. Because most of our instructors have little to no familiarity with GitLab or DVCS in general, we provide instructors with a faculty and staff handbook during on-boarding. The handbook includes a variety of DVCS and Courseware as Code topics such as our project repository structure for courses, our access request process, and how-to guides for git. Depending on the expertise of an incoming instructor, we may assign them under a module manager. The module manager will integrate the instructor by providing training on syntax, format, commands, and answering individual questions. Fully integrated instructors can contribute as content developers at any time and from any location. We conduct most development discussions within the applicable GitLab course project through commit comments, merge requests, and issues. Instructors can develop and discuss content without depending on a set weekly or monthly change meeting. The DVCS automatically stores historical data of what changes were made, when they were made, who made them, and what discussions occurred. Demonstrating the value and utility of our automated pipelines and our interest in their input helps create a desire to learn in the majority of our instructors. When launching, we provided training classes and meetings to present and discuss the concept and address specific workflow nuances. Our instructors have understood the concept best through consistent leadership in requiring them to use the DVCS content and processes and pure repetition. Several instructors initially resisted and stated their need of WYSIWYG editors, but the power of universal by-line version control for the entire history of every piece of courseware overcame their hesitance and concerns by providing peace of mind in the change process.

IV. Courseware as Code results

The total number of commits since inception provides the most objective indicator of Courseware as Code's success (see Figure 1). Although we understand the number of commits does not directly correlate to the amount or quality of content added and changed, it provides one metric by which to gain an appreciation of scope. Over our inagural year of using Courseware as Code from January 2017 to January 2018, our school taught over 250 students in 30 different classes that included the core, C, bash, Powershell, and Python courseware.

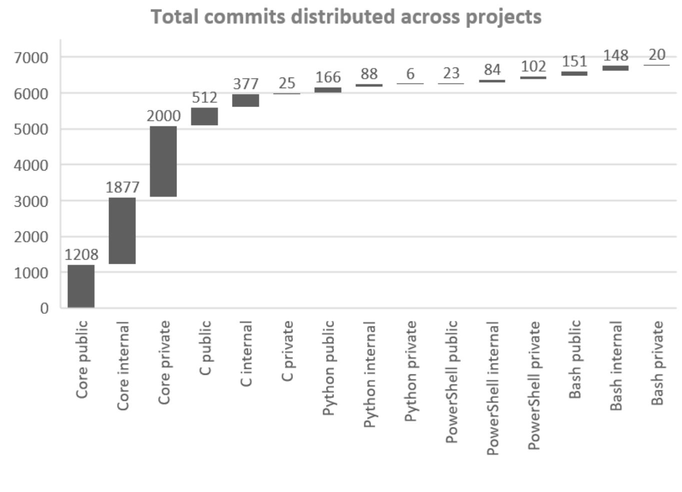

Fig. 1. Total repository contribution in commits from January 2017 to April 2018 by project.

By April 2018, our courseware base had 6,787 commits over 15 projects with an average of 26.5 commits per instructor. Objectively with these numbers and subjectively with observed and stated instructor and student material engagement, we believe we have exceeded simple compliance and achieved true buy-in to Courseware as Code. For our core course, 29 instructors over up to three disparate class locations stayed engaged with 5,085 commits and 113 issues raised and discussed (see Figures 1 and 2). Our oversight for the programming courses allows greater autonomy and variance in course delivery and assessment than we require for our core course. This results in less need for synchronization and discussion, as evidenced by fewer issues raised (see Figure 3). Despite less discussion, each of the programming courses have over 200 commits between their three projects with C programming leading the pack at 914 total commits. Subjectively, the value in providing new instructors with not only course material but also the history of courseware development has enabled instructor turnovers to occur with little to no dips in course quality. Instructors have embraced the DVCS peer-review process. The ability to view courseware and its evolution on a by-line basis enables instructors to better understand gaps between what they observe for actual classroom outcomes in contrast to intended learning outcomes. This also provides instructors and all contributors with the freedom to independently develop innovative changes to courseware and submit their ideas and content forward for review, approval, and implementation. Even if a submission does not get approved, it sparks a discusson on technical content and andragogy that can benefit the entire faculty both at that time and in the future.

As a whole, our GitLab instance has synchronized over 175 users with an average of 6.7 merge requests submitted per user. Out of these users, over 100 of them are not in our faculty. These external users include students and workforce members. While some of these users have contributed to our courseware, many opt to use the platform for their own small team organizational training based on its efficacy for accountable

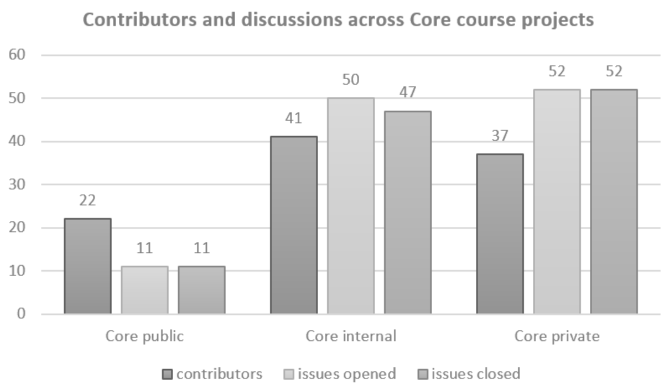

Fig. 2. Core course participation in number of total contributors from January 2017 to April 2018, number of issues opened, and number of issues closed.

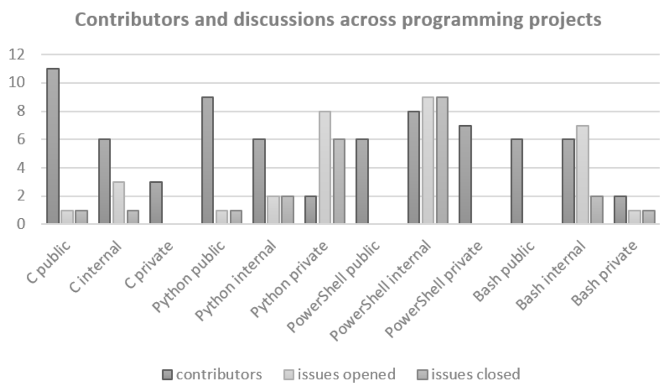

Fig. 3. Programming participation in number of total contributors from January 2017 to April 2018, number of issues opened, and number of issues closed.

collaboration, synchronization, and iterative development.

V. Conclusions about Courseware as Code

Based on the demonstrated success of using DVCS for courseware management, synchronization, and improvement, our school will continue using it to maintain our technical course content. Future growth could include opening it up as a platform for student submissions as well as per-question granularity in tracking assessment performance. Since its inception, we have seen nearly 7000 commits to courseware contributed by over 41 authors including 9 student authors. Courseware contributors raised over 140 issues of which over 130 were resolved. We believe this high amount of participation proves the utility of the Courseware as Code concept for individual and small group educational content management and iterative improvement. Although we used GitLab, a DVCS primarily used for computer code version control and the creation and maintenance of computer-related curricula, the benefits we have illustrated could also prove useful for non-technical curricula. The ability to crowdsource courseware development and co-opt distributed experts exponentially increases capacity and broadens and deepens the capability of curriculum development teams. Additionally, a distributed core of subject matter experts can rapidly review courseware that requires dynamic revisions for a variety of reasons (e.g., advancements in technology, technology obsolescence, evolution of business practices), further enabling an efficient CI/CD process with rigorous approvals. We recommend other educational institutions consider DVCS and automated workflows to enable transparency in materials and organized synchronization that empowers widespread contribution to further education.

References

[1] Curtis Clifton, Lisa C. Kaczmarczyk, and Michael Mrozek. 2007. Subverting the fundamentals sequence: using version control to enhance course management. In Proceedings of the 38th SIGCSE technical symposium on Computer science education (SIGCSE '07). ACM, New York, NY, USA, 86-90.

[2] J. Kelleher, "Employing git in the classroom," 2014 World Congress on Computer Applications and Information Systems (WCCAIS), Hammamet, 2014, pp. 1-4.

[3] Joseph Lawrance, Seikyung Jung, and Charles Wiseman. 2013. Git on the cloud in the classroom. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). ACM, New York, NY, USA, 639-644.

[4] S. Mandala and K. A. Gary, "Distributed Version Control for Curricular Content Management," 2013 IEEE Frontiers in Education Conference (FIE), Oklahoma City, OK, 2013, pp. 802-804.

[5] A. Tirkey and K. A. Gary, "Curricular change management with Git and Drupal: A tool to support flexible curricular development workflows," 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA), London, 2017, pp. 247-253.

[6] A. Zagalsky, J. Feliciano, M. Storey, Y. Zhao, and W. Wang. "The Emergence of GitHub as a Collaborative Platform for Education," Motivation and Dynamics of the Open Classroom, CSCW 2015, Vancouver, BC, Canada, March 14-28, 2015.