Courseware as Code

Setting a new bar for transparency and collaboration

Julianna Rodriguez*, Christopher Apsey†, Sarah Rees‡, Todd Boudreau†, George Raileanu§
*U.S. Army Cyber Command, Fort Gordon, Georgia 30905, julianna.m.rodriguez3.mil@mail.mil
†U.S. Army Cyber School, Fort Gordon, Georgia 30905, christopher.w.apsey.mil@mail.mil, todd.m.boudreau.civ@mail.mil
‡Georgia Cyber Center for Innovation and Training, Augusta, Georgia 30901, srees@augusta.edu
§U.S. Army Cyber Protection Brigade, Fort Gordon, Georgia 30905, george.r.raileanu.mil@mail.mil

***Abstract*— This Innovative Practice Category Work In Progress presents a case of creating and managing courseware as code. By managing all content for five central courses in machine-readable markup language within a GitLab instance, our school achieves rapid synchronized collaboration. Through GitLab, we facilitate change discussions, provide transparency in updates, and allow faculty, students, and even our 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 cyber technical 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, the Secretary and Chief of Staff of the Army established the Cyber branch and the U.S. Army Cyber School (USACYS). Building USACYS from scratch required us to build courseware to fill workforce-oriented curricula. Before we could finish creating sufficient and suitable content, the Army assigned students and scheduled classes. As we worked to rapidly create content (often described as 'building an airplane while in flight' by our commandant), we turned to cloud-hosted storage and synchronization options. However, we found the options for simple storage of non-markup language files could not support a managed workflow of creation, sharing, review, and approval. With many contributors but without control on groups, permissions, and content visibility and acceptance, we recognized our need for increased structure and improved workflows. We had already chosen to create a privately hosted GitLab instance to hold our virtual training infrastructure configuration code for its ability to track each change on a line-by-line basis through commit history logs. Through this GitLab instance, we also chose to implement an 'Everything as Code' mentality to our courseware.

II. Related Work

We could not find a documented courseware synchronization case that matched our scope when we researched literature for using distributed revision control systems (DVCS) in educational courses. 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 [5]. Continuing in 2017, ASU proposed a new web framework with a GitHub backend in which instructors use markdown through SimpleMDE [6]. As before, they did not use their proposed system in a class, settling instead for a faculty review of the framework's potential. Instructors of a 2015 web software development class at Aalto University used GitLab for disseminating course material to 225 students [2]. They managed all of their material through the DVCS to include lecture slides generated from HTML. The students overwhelmingly appreciated accessing material through Git and actively submitted corrections when they identified errata.

Other published cases have 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 [4]. 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 [3]. 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 [7]. The authors of that research identified limitations in DVCS adoption due to a lack of

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

support for PDFs and LaTeX and no existing measurements of student contribution or efficacy on fostering a participatory culture.

Although related, none of these cases included more than a few instructors in one location and on a time-synchronized schedule, whereas our case includes up to 29 instructors in different physical locations and different points in the courses. Through 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.

III. Courseware as Code concept

Using a foundation of DevOps and Continuous Integration/Continuous Delivery (CI/CD) philosophies, our idea for Courseware as Code extends the idea of "Everything as Code" to educational content. Leveraging applicable aspects of the software development lifecycle to courseware facilitates discussions of suggested changes amongst faculty, provides transparency in updates, enables custom workflows based on the complexity of course content, and allows faculty, students, and workforce members to identify issues and contribute content.

The Courseware as Code concept requires replacing traditional document formats (e.g. PowerPoint slides, Word and other word processor documents, spreadsheets, and other binary files) with markup language formats and leveraging CI pipelines to create PDFs, html5 slide decks, and more for distribution. Markup language formats can include Asciidoctor, markdown, LaTeX, reStructuredText, or any other markup based formatting and preparation method. These non-binary formats allow for text parsing and comparisons via change logs for review, discussion, and approval. This fine-grained control promotes trust in the integrity of the material and enables instructors to track every content change.

Using a DVCS such as GitLab to manage markup language documents allows creating 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 can support zeroing in on specific commit sets to analyze whether a specific change improved or degraded student performance. Applying this idea directly to assessments makes test material validation possible on a per-question basis. By organizing each question with metadata providing objective correlation, workflows that automate test assembly can choose questions randomly from an objectives-based associative array. Contributors can write test questions and exercises alongside stack infrastructure orchestration templates (e.g. Heat YAML files) from which students can launch sandbox exercise environments. Designated faculty can approve new content and changes through branching, merging, and rolling back merges when needed so as to enable maximum flexibility in creative freedom. Courseware as Code opens up many possibilities for efficient and effective collaboration, content creation, and substantive change management.

IV. Courseware as Code implementation

USACYS 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 Cyber Common Technical Core (CCTC). CCTC engages students in problem solving, research, and understanding 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 of user groups for encouraging collaboration while maintaining accreditation-required controls and material integrity.

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, automating workflows, and instructor buy-in. Since launching in January of 2017, our implementation has gained momentum and successfully guided instructors in collaborative content creation.

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 designed an account creation and approval process that leverages the DoD PKI infrastructure. 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. 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'. We require non-faculty contributors to sign a contributor license agreement and, if working with internal or private projects, 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

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

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

branches based on 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.

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. While we currently support version control for Heat templates, future pipelines could also include automating Heat template deployments of OpenStack infrastructure resources for student labs and assessments.

For Courseware as Code to be successful, we had to communicate the vision to our entire faculty. Most of our instructors have little to no familiarity with GitLab or DVCS, so 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 for changed contents, including 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 guidance 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 every piece of courseware overcame their hesitance and concerns by providing peace of mind in the change process.

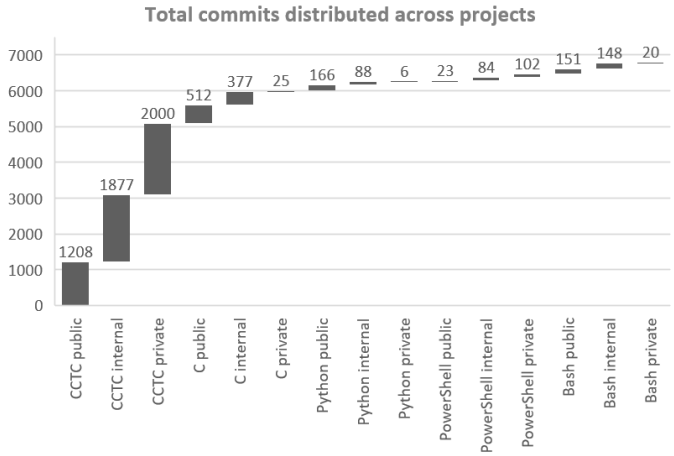

Fig. 1. Total repository contribution in commits from January 2017 to April 2018 by project.

V. Courseware as Code results

The total number of commits since inception provides an 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 inaugural year of using Courseware as Code from January 2017 to January 2018, USACYS taught over 250 students in 30 different classes that included CCTC, C, bash, Powershell, and Python courseware. 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 basic compliance and achieved buy-in to Courseware as Code. For CCTC, 29 instructors over up to three disparate class locations in separate states 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 better enables instructors to 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 or

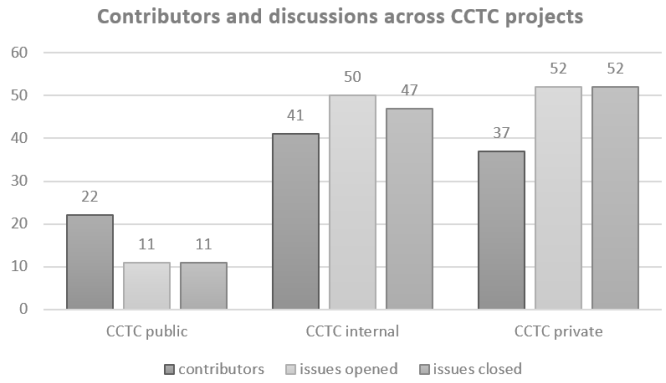

Fig. 2. CCTC participation in number of total contributors from January 2017 to April 2018, number of issues opened, and number of issues closed.

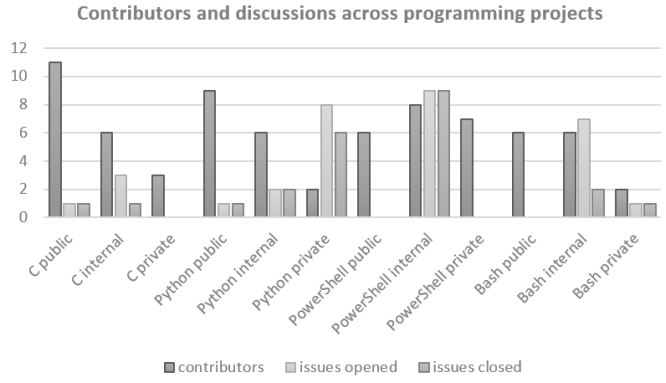

Fig. 3. Programming participation in number of total contributors from January 2017 to April 2018, number of issues opened, and number of issues closed.

content forward for review, approval, and implementation. Even if a submission is not approved, it sparks a discussion 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 USACYS 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 collaboration, synchronization, and iterative development.

VI. Conclusions about Courseware as Code

Based on the demonstrated success of using DVCS for courseware management, synchronization, and improvement, USACYS will continue using it to maintain our technical course content. Future growth could include utilizing it for student submissions as well as per-question granularity in tracking assessment performance. Since its inception, USACYS has 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 the GitLab DVCS we used primarily supports computer code version control and the creation and maintenance of computer-related curricula, the benefits we illustrated could also improve non-technical courseware synchronization and development. The ability to crowdsource courseware development and co-opt distributed experts exponentially increases capacity and 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.

VII. Acknowledgment

The authors would like to thank Samuel Anderson, Natasha Orslene, Dan Poulin, Jessie Lass, Edward Woodruff and Tim Nosco for helping build a foundation on which the Courseware as Code concept could succeed. Their help in writing content and setting workflows, processes, and expectations enabled this concept to become our standard. We also thank BG Jennifer Buckner and Mr. Tom Barnes for approving Courseware as Code to become a reality and believing in this vision to set USACYS up for success both in the Army and in the academic and technical education communities.

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] Lassi Haaranen and Teemu Lehtinen. 2015. Teaching Git on the Side: Version Control System As a Course Platform. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '15). ACM, New York, NY, USA, 87-92.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.