A Proposed Workflow For Version-Controlled Assignment Management

Bob Edmison
Virginia Tech Department of Computer Science
Blacksburg, Virginia, USA
bedmison@vt.edu

Austin Cory Bart
University of Delaware Department of Computer Science
Newark, Delaware, USA
acbart@udel.edu

Stephen H. Edwards
Virginia Tech Department of Computer Science
Blacksburg, Virginia, USA
edwards@cs.vt.edus

ABSTRACT

Computer science instructors spend a significant amount of time developing software exercises and projects for their classes. After creating these assignments, sharing them with their colleagues can be another large effort, even at the same institution. As part of an effort to scale course offerings, as well as sharing content, we propose a solution to leverage two existing technologies, Waltz and PEML, to create a complete workflow that will allow an instructor to store their programming assignments in a Git code repository, use PEML to define the exercise, and then use Waltz to create the student-facing resources in a learning management system (LMS), as well as the automated creation of program assessments in autograders. We will also discuss future opportunities to extend this workflow.

Author Keywords

automated grading, programming assignment, interchange, reuse, SPLICE, PEML

CCS Concepts

•**Applied computing** → **Computer-assisted instruction;** •**Social and professional topics** → **Computing education;**

INTRODUCTION

Developing programming assignments is a labor-intensive activity, especially so when an instructor is teaching a course for the first time. Many instructors are willing to share the exercises they create, but the manner in which those assignments are shared also can be labor-intensive [6].

Attempts have been made to create repositories of programming assignments [12][11] and other work has been done on how to design a resource sharing site [10]. Indeed, a format for encoding the metadata that describes programming exercises has been developed (PEML [4]). The missing link is a pipeline to take materials from a repository, publish them to a student-accessible location, and configure the automated grading tools that assess the students' submissions of those projects (ideally in a automatic way, as in CI/CD). We describe a potential solution that leverages work done by several of the CS SPLICE working groups [3][4], incorporating technology and best practices developed by those groups.

Seventh SPLICE Workshop at SIGCSE 2021 "CS Education Infrastructure for All III: From Ideas to Practice", SPLICE'21, March 15-16, 2021, Virtual Event

MINIMUM WORKFLOW SUPPORT

As with most things, lowering the barriers to entry makes it more likely that a tool or process will be more widely adopted. To do this, we believe that a workflow would need the following minimum features:

- *Use a standard, machine-readable format to describe the programming assignment.* There are a variety of ways in which a programming assignment can be described. Many of the existing tools and assignment repositories leave it to the contributor to format the metadata that describes the assignment. This can result in uneven or incomplete descriptions of the materials. This approach can also hinder automatic conversion of the materials into other forms, because of the ad hoc nature of the description.

- *Interface with an LMS to host the student-facing assignment content.* The mechanics of creating the student-facing artifacts for programming assignments can be tedious. Some of the metadata for an assignment, such as due dates, are independent of the specific instance of offering the assignment. Attempts have been made [13] to develop workflows to automatically create the student-facing artifacts deployed separately from an LMS. Because of the prevalence of LMS tools, as well as the reported preference of students to have a central location for course materials [8], interfacing directly with an LMS to create the student-facing artifacts is viewed as a critical feature.

- *Be able to read project resources from a version control repository.* Using a version control repository to hold the course materials provides a number of benefits. Such a repository provides a built-in mechanism for tracking changes and additions to the assignment collection. Additionally, sharing is facilitated by allowing access to the repositories to invited instructors, who can make additions and modifications to assignments. Finally, versioning

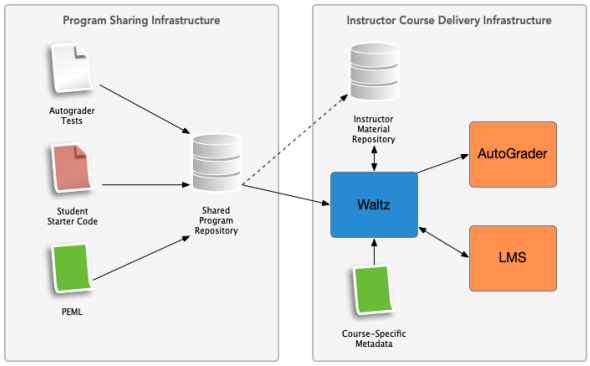

Figure 1. Workflow Overview

can be used to identify the edition of the assignments being offered. The use of Github and Gitlab as homes for course material repositories is well understood, with some instructors hosting their entire course in Github, and then forking an instance to use as the basis for the next term [13].

- *Interface with a widely-used automated grading platform to configure the process of assessing the projects.* Publishing programming assignments to students is often only half of the necessary preparation. Instructors must also configure their autograder to accept and evaluate student coding submissions. The evaluation scripts or tests should be captured with the assignment, and the publishing workflow should be able to push those configurations to the autograder.

TOWARD A COMPLETE WORKFLOW

Our goal is to incorporate all of the features specified in Section 2 into a single tool. We propose a solution that leverages existing tools to create a complete workflow between the code-repository, LMS, and autograder. As a start, the assignments will be published in a Gitlab [7] repository hosted by the first author's university. This repository uses a federated authentication system to allow access to contributors from over 500 institutions. Initially, access to the repository will be limited to the project participants. Eventually, the intent is to grant access to verified instructors who wish to use the assignments provided, or contribute their own. In addition to the value provided by federated authentication, we opted for a locally-hosted repository because we believed this offers an extra layer of security to ensure the integrity of the course resources, while offering all of the features of Git. As this community grows, we envision a catalog of supported repositories being available, with indexing based on the tags provided in the the project metadata.

We will use the Programming Exercise Markup Language (PEML) [4] to capture instance-independent metadata for the assignments. Each assignment will have a PEML file that describes the assignment. We will capture metadata such as the assignment author, the topics covered in the assignment, as well as any prerequisite knowledge the author believes students would need prior to undertaking the assignment. Also, PEML provides a way to reference supporting resources for an assignment, such as starter code and test cases, as well as a reference implementation and tests for use in an autograder. We have created PEML files for each of the assignments in the repository. As part of this effort, we will create a publicly-accessible microservice that will validate and provide a parsed version of the PEML file. These files will be the primary input into the workflow tool.

Waltz [1] will be used to manage the workflow process. Waltz is a command line tool created in Python. It will create the student-facing assignment resources from the PEML files and referenced resources, and then publish them to an LMS. Initially, LMS support will focus on Canvas [9], as this is the LMS used at the authors' institutions. Additional LMS support is intended, as we find contributors who use to Blackboard [2] and D2L [5]. Additionally, Waltz will manage the configuration of the autograder, based on the configuration files described in the assignment PEML file.

Finally, this process will automatically provision resources within supported autograders. Initially, we will support Web-CAT as the autograding component of this workflow. Support for additional autograding tools will be added as demand for those services become clear. By parsing the PEML file, Waltz will be able to retrieve the reference tests to be used to assess student responses to the programming assignment. The process will then reach into supported autograders to programmatically configure of the grading infrastructure. PEML files can support an arbitrary set of autograding configurations. Our initial support is focussed on Web-CAT for two reasons: It has been developed and maintained by one of the co-authors since inception, and; Web-CAT supports LTI integration into Canvas, thus we will be able to establish the linkage between Canvas and Web-CAT to allow submissions from inside the Canvas Assignment tool and also allow Web-CAT to pass grades back into the Canvas Gradebook once the assessment is completed. This provides a complete end-to-end submission and grading workflow.

FURTHER WORK AND CONCLUSION

As we execute on our vision, we are looking for three major things from the community. First, we hope for feedback on the problem and solution we have identified; what limitations and issues do you foresee? Our goal is for this workflow to be as flexible as possible, but not unwieldy in its flexibility. Second, we are interested in suggestions for new features or extensions to this work that might be of interest to the community. While we plan to add features such as additional LMS and autograder support, we would like to know which other tools to support first. Finally, we are advertising our workflow in the hopes of finding potential collaborators and adopters, to increase our impact. We would very much like to identify instructors who are interested in extending this work, either by contributing to the development of the tooling, by using the tool and giving us feedback on what works well, as well as the pain points that are encountered, and also to help us identify shortcomings in the tools and processes. We hope to make this a solution that contributes to reducing the effort needed to share resources, as well as deploy them for our students.

REFERENCES

[1] Austin Cory Bart. 2021. Waltz: A Software System for Synchronizing LMS Course Content. https://github.com/acbart/waltz. (2021). Accessed: 2021-02-09.

[2] Blackboard. 2020. Blackboard Learning Management System. (2020). https://www.blackboard.com/teaching-learning/learning-management

[3] CS SPLICE Curriculum Materials Working Group . 2021. Curriculum Materials Working Group. https://cssplice-cm.github.io/. (2021). Accessed: 2021-02-09.

[4] CS SPLICE PEML Working Group . 2021. PEML: Programming Exercise Markup Language. https://cssplice.github.io/peml/. (2021). Accessed: 2021-02-09.

[5] Desire2Learn. 2020. Brightspace LMS For Higher Education. (2020). https://www.d2l.com/higher-education/ Section: Adaptive Learning.

[6] Stephen H. Edwards, Jürgen Börstler, Lillian N. Cassel, Mark S. Hall, and Joseph Hollingsworth. 2008. Developing a Common Format for Sharing Programming Assignments. *SIGCSE Bull.* 40, 4 (Nov. 2008), 167–182. DOI: http://dx.doi.org/10.1145/1473195.1473240

[7] Gitlab, Inc. 2021. Gitlab. https://https://about.gitlab.com/. (2021). Accessed: 2021-02-09.

[8] Nazire Burcin Hamutoglu, Orhan Gemikonakli, Ibrahim Duman, Ali Kirksekiz, and Mubin Kiyici. 2020. Evaluating students experiences using a virtual learning environment: satisfaction and preferences. *Educational Technology Research and Development* 68, 1 (Feb. 2020), 437–462. DOI: http://dx.doi.org/10.1007/s11423-019-09705-z

[9] Instructure. 2018. Learning Management System | LMS | Canvas by Instructure. (Dec. 2018). https://www.canvaslms.com/

[10] Mackenzie Leake and Colleen M. Lewis. 2017. Recommendations for Designing CS Resource Sharing Sites for All Teachers. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 357–362. DOI: http://dx.doi.org/10.1145/3017680.3017780

[11] National Center for Women and Information Technology . 2020. EngageCSEdu. https://www.engage-csedu.org. (2020). Accessed: 2021-02-09.

[12] Nick Parlante. 2020. Nifty Assignments. http://nifty.stanford.edu/. (2020). Accessed: 2021-02-09.

[13] Phill Conrad. 2021. University of California Santa Barbara CS Course Repository. https://github.com/ucsb-cs-course-repos/ucsb-cs-course-repos.github.io. (2021). Accessed: 2021-02-09.