Version Control Open Source Software for Computer Science Programming Courses

Mesafint Fanuel[1], Tzusheng Pei[1*],
Ali Abu El Humos[1], Xuejun Liang[1] and Hyunju Kim[2]
[1]Department of Computer Science
Jackson State University
Jackson, MS 39217
{mesafint.d.fanuel@students.jsums.edu}{tzusheng.pei, ali.a.humos, xuejun.liang}@jsums.edu
* Corresponding author
[2]Department of Mathematics and Computer Science
Wheaton College
Wheaton, IL 60189
hyunju.kim@wheaton.edu

***Abstract*—The programming experience gained in a computer science programming course is hugely inadequate in creating the next generation of capable developers. This is because the small lines of code students write in class are not demonstrative of software development as it occurs in industry. Thus, most computer science students lack critical skills needed to be capable developers. In addition to that, teamwork is crucial in the development of new software or the maintenance of old systems. This project has built an in-house version control system using only open source products to provide students with the platform to experience collaborative development. This educational environment will instruct students on Open Source Software (OSS) community. It will familiarize students with writing large scale software code and introduce them to version control tools hugely utilized in software development. Students will experience the various aspects of software development by playing different roles while allowing instructors to easily track student activities. Consequently, students can reuse the code and artifacts as examples or basic frame works for their development. Coding projects progress can be easily tracked by the instructors and team leaders.**

Keywords—version control, software development, Open Source Softwar, GitLab

I. INTRODUCTION

A major challenge in teaching software engineering is the gap between real development activities and the materials taught in class. Being highly invested in the theoretical, most of software engineering education doesn't prepare students for the industry. Students will be introduced to the reality of software development the hard way once they graduate. The best way to tackle this mismatch is through skill oriented educational strategies. Programs students write in class settings, being few hundred lines in length, are not typical in real world development. Students need to be made proficient in writing large scale software products through team work (collaboration).

Students need not only lectures but an experience in problem modeling, algorithmic thinking, collaborating with team members, and following the software development life cycle. They need to be induced to put in thousands of hours of coding practice to master the trade of software developer. This paper reports our effort to adopt an OSS community model to educate CS students in software engineering, specifically collaborative software development. We built a client-server system to support students' software development and team activities. Students can learn from real-world code examples and team dynamics by participating in OSS projects that are hosted at the community system.

II. BACKGROUND

Open source software tools have been used in CS education in order to teach traditional or online CS courses, such as programming, data structures, algorithms, and software engineering [2, 8, 9]. Although such previous practices simply used OSS tools to teach the subjects, they demonstrated that the use of OSS in classrooms resulted in positive impacts.

In addition to using OSS tools several studies have also brought OSS projects to the classroom [3, 4, 10, 12]. Finding have shown that OSS projects provide students a unique opportunity in developing software in a real-world environment. By participating in these projects, students not only sharpen their coding skills, but also learn how to work with teams. They also become more familiar with intellectual property, software licensing issues and acquire working knowledge on the domains the project revolves around [1].

Nevertheless, teaching Computer Science students software engineering using open source projects brings a set of unique challenges to the classroom. The code bases of open source projects as well as the tools used to collaborate on them are complex. Students will not have the time to make concrete contributions to a project in academic terms since it will take weeks for them to even get well informed about the project.

Setting up a development environment suitable to everyone can be complicated in light of varying degrees of familiarity students have to tools and platforms. Open source Communities also have particular development methodologies and norms that one can learn only through extensive participation in them. Furthermore, class academic calendars may not overlap with a projects release schedules. This makes joint work even more difficult.

Thus, instead of embedding students into preexisting communities it is sensible to let students create their own communities which could start out as assignments posted by the instructor. This assignment could be month long codes. And by giving software engineering courses for two semesters or more, or by crafting the appropriate educational model, students will be able to deliver dozens of large scale (1000 or more lines) of OSS software per year. This will greatly increase their coding and team work skills. Instructors will give regular grades to the students by checking the work being done on the Open source community. This paper reports our effort to establish such a teaching environment called Student-Source for such OSS benefits while minimizing the identified limitations.

III. THE STUDENT-CENTERED OPEN SOURCE SOFTWARE (SOSS) COMMUNITY

A fundamental paradigm shift in software engineering education will be the unintended outcome of Student-Source. By allowing students to work on semester long—even year round—projects, grades will escape the confines of the traditional classroom. By default, students will have a plethora of project code and artifacts accumulated at a central location. Thus, the community system acts as a virtual classroom, and students learn by examples and from peers. The grade they will gate will count to their senior project or to an overall course requirement.

Furthermore, Student-source will afford a different standard of instructor to student interaction. A CS student's education will not jus be theoretical but experiential. The traditional confines of the classroom will be supplemented by development activities on the community system. As students learn more throughout the year they will add a little to the OSS projects they are undertaking on the system. Students can also work on the various OSS projects taking place on the system as well.

Most development of OSS happens in university initially. The tool will make the university a hub of OSS development activity. Many projects will be started. It will be an enabler in creating a microcosm of the real software industry in the university setting. Overall Student-Source will provide active learning environment, which allows collaborative learning, intuitive learning, role playing, etc.

A. Structure and Functionalities

The front end of Student-Source contains a web site that allows user to log in or create a user account through Google OAuth authorization protocol. We implemented the Google OAuth authorization protocol along with the restriction that the domain must match one of the institution's email domains. Google OAuth gives users the ease of creating accounts automatically by just signing into our institutional email system that is supported by Gmail. These functionalities keep the scope of Student-Source within the confines of the university.

Another way we created a controlled teaching environment is by disabling public visibility level of Student-Source. Student-Source offers three visibility levels on project repositories. By visibility level we mean permission types to one's code and project artifacts. This visibility types are private visibility meaning repos are available to users explicitly given access, internal visibility gives access to any user logged in and public visibility gives access to anyone for free.

All users will be categorized as either admin or student. Each category possesses different permission levels that can access diverse functionalities of Student-Source. Students have traditional permission and control just like any conventional user of version control tools such as GitHub. They can clone project, pull projects, collaborate on projects and so forth. A list of student activities on the system appears on the student dashboard page. Various other features exist on Student source web frontend as well making the system a highly integrated and rich platform for users.

Admin users have an overall view of the system from their admins control panels. They can post announcements on Student-Source through the announcement feature. Admin rights can be given to instructors so that they can easily track everything that is happening. In future works, we seek to dissect admin and create a new user role called instructor with its own permission levels and advanced functionalities. Already our team has implemented a faculty forum page for instructors on the system.

On the backend, Student-Source is hosted on a Nginx web server, and this in turn runs on an Ubuntu 14.04 host machine. The community users can access their project repositories by using Git client or EGit, which is supported by Eclipse. EGit works with JGit (the Java implementation of Git). Currently, the server is running on a generic desktop with a quad-core processor and 4GB RAM. Student-source functionality is backed by a PostgreSQL database.

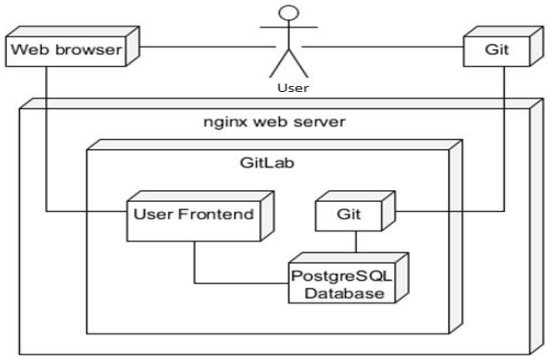

Fig. 1. The SOSS server architecture.

B. Open sources software's used

Built on top of the famous web-base Git management tool, GitLab, our Student source contains all the rich features of these management tools. These features include issue tracking, built in wiki systems, protected branches, code snippets, project importing and unlimited public and private repositories. Projects can be organized into private, internal or public and access to them can be managed with five different users setting for external users.

We decided to use GitLab because it is Git server based upon the famous source code management tool, Git. Of all the source code management tools out there such as subversion, mercurial, we found Git to be by far superior. It is the default version control system for large open source repository websites such as GitHub and Bit Bucket. Previous surveys have also shown Git to be the most preferred version control tool.

The benefits of Git include a full history of code changes so that a user can easily pull previous versions of the code, and a branch mechanism so that the user can develop and test different scenarios on the code. Additionally, its repository-to-repository interaction allows the user to share entire branches between repositories.

IV. Utilziing Student-Source

The SOSS Community system was first introduced graduate and undergraduate software engineering courses during Fall 2014 semester in the Department of Computer Science at Jackson State University. In subsequent years we have introduced the system to lower level CS and CE classes at our institution. These efforts have brought to light challenges that need to be tackled before a complete adoption and a successful application of student-source in Software engineering.

As identified earlier, it is challenging to get the student body used to the system and its corresponding collaboration tools. Currently our team uses a series of modules to be distributed in classes during our brief introductory sessions. These short modules contain a clear step by step guide on how to get started on Student-Source. While students have overall been successful in following guidelines, the experience has highlighted the need for students to grasp concepts related to Student source more successfully as a prerequisite.

Our team realizes the necessity of having deeper introductory sessions to teach students Git and basic CLI commands. An initial introduction of Git and CLI must also be followed by a short phase of familiarizing students with them. Such sessions could be held in programming labs in order not to require fundamental changes to curriculums [13]. Furthermore, these sessions specifically and student-source in general must be introduced to lower-level CS courses (e.g. programming course) so that students can utilize the system throughout the course of their CS education.

Secondly it is arduous to build a dynamic community full of open source projects and usable OSS products based purely on student initiative. An open source community needs leaders, those who initiate the projects and oversea the various projects. We have seen the tremendous role instructors will play in this area. There unique position allows them to easily be the instigators of various open source projects. By lightly integrating Student-Source into their curriculum as a supplemental material, instructors play the leading role in building up dynamic communities.

While the overall impact of our efforts had been positive, we have discovered the traditional road block to a complete adoption of the community system. A community needs leaders and the instructors are the initiators of projects on student source. We propose an educational model were instructors include Student-source as a supplemental material to there curriculum. These will motivate students to play active role in the community and build up various OSS products.

V. Conclusion and Future Work

Moving forward, we plan to introduce a third category of users to the system, Instructors. The instructor user will make student source more fit for the educational environment. The instructor will be afforded a whole range of actions such as checking student accounts and tracking student activities.

Instructors should be able to grade students based on various development and collaborative activities students do on the system. A grading mechanism will grade students based on the number of commits, number of insertions in the commits, number of deletion in the commits, and total line of code written by student as they are extracted from the total number of commits done by the students. Other factors such as number of collaborative task such as pulling project, adding students, commenting on work and so forth will also be factored out by the grading methodology. In conclusion, the admin controls the system; the students collaborate on the system, while instructors track student activities on the system.

Acknowledgment

This work has been supported through the National Science Foundation grant (HRD-1348565) on the SOSS (Student- centered Open Source Software) Community. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the funding agency.

REFERENCES

[1] Bryan Behrenshausen, "Professors embed students directly into open source communities", http://opensource.com/education/14/8/challenges-to-open- source-computer-science-education, 2014.

[2] D. Carrington and S. Kim, "Teaching software design with open source software", In Proc. of the 33rd ASEE/IEEE Frontiers in Education Conference, 2003, pp.9-14.

[3] R. Charles and Y. Tao, "Evaluating student participation in open source software development with an annotation model", In Proc. of the 4th IASTED International Conference on Knowledge Sharing and Collaborative Engineering, 532-063, 2006.

[4] H. Ellis, M. Purcell, and G. Hislop, "An approach for evaluating FOSS projects for student participation", In Proc. of the 43rd ACM Technical Symposium on Computer Science Education, 2012, pp.415-420.

[5] Git, http://git-scm.com.

[6] GitLab, https://about.GitLab.com.

[7] Google Developers, "Using OAuth2.0 to access Google APIs", https://developers.google.com/accounts/docs/OAuth2.

[8] S. Lakhan and K. Jhunjhunwala, "Open source software in education", Educause Quarterly, vol. 31, no. 2, 2008, pp.32-40.

[9] D. Lipsa and R. Laramee, "Open source software in computer science and IT higher education: a case study", International Journal of Advanced Computer Science and Applications, vol. 2, no. 1, 2011, pp.43-54.

[10] Antoine Melki, "Proprietary versus open source software in support of learning in computer science", In Proc. of the 2014 Federated Conference on Computer Science and Information Systems, 2014, pp.111-115.

[11] Ian Skerret, "Eclipse community survey 2014 results", https://ianskerrett.wordpress.com/2014/06/23/eclipse- community-survey-2014-results/, 2014.

[12] S. Sowe, I. Stamelos, and L. Angelis, "An empirical approach to evaluate students participation in free/open source software projects", In Proc. of IADIS International Conference on Cognition and Exploratory Learning in Digital Age, 2006, pp.304-308.

[13] L. Vu, T. Tan, and P. Maneerat, "Incorporating open-source software development into computer science and software engineering education at university level", In Proc. of the 2nd Australian Undergraduate Students' Computing Conference, 2004, pp.149-164.