Using Git to Manage Capstone Software Projects

An Empirical Research Report

Zhiguang Xu
Department of Math and Computer Science
Valdosta State University
Valdosta, GA, USA
zxu@valdosta.edu

***Abstract*—Distributed software project development has become a reality not only in industry but also in computer science classes nowadays – students and teachers have to leverage time, talent, and resources collaboratively wherever they reside, especially when everyone is working on his/her own schedule, from his/her convenient location, and using various programming systems. In this paper, we will present an empirical study of how *Git*, "a free & open source, distributed version control system", is used in an undergraduate Computer Science (CS) capstone class to facilitate team collaboration for the students and to ease the project assessment and grading tasks for the teachers. Other Git-related aspects such as preventing plagiarization, hosting online public/private project repositories, and improving the student-teacher interactivity during lecture sessions, are also discussed. Despite of the relatively bumpy and steep learning curve in the beginning of the semester, all four groups of students in the capstone class described in this paper benefitted tremendously from Git, which reduced the burdens of version control and group management on their shoulders, increased the collective productivity of their groups, and helped them in completing their substantial software projects successfully. This paper is concluded with a vision on expanding and standardizing the adoption of Git in other Computer Science classes in the future.**

***Keywords - Distributed Student Software Project Management; Distributed Version Control System: Git; Computing and Information Sciences Education*.**

I. INTRODUCTION

CS 4900, *Senior Seminar*, is a project-driven course designed to provide senior capstone experiences for graduating Computer Science majors at Valdosta State University (VSU). In fall of 2011, twelve students in this class formed four groups to write full-fledged *Ruby on Rails*-based Web server applications that were accessible not only from regular Web browsers but also from *Android* mobile clients that they developed.

In Section II, we present reasons why a *Distrusted Version Control System (DVCS)* is very much needed in CS 4900 and what features that it ought to have. Then, in Section III, we will provide a literature survey of popular DVCSs (Git being one of them) under the umbrella context of Collaborative Development Tools. This is followed by Section IV, an in-depth review of how a Git system is setup, configured, and used in CS 4900. Then, in Section V, three workflows with Git are presented to show what kind of services Git (configured in the way as described in Section IV) provides to students and teachers to increase the overall productivity of the whole class. Finally, this paper concludes with Section VI, a vision on and future works planned for expanding and standardizing the adoption of Git in a wider range of Computer Science programming classes.

II. BACKGROUND

In this section, we will provide the *pedagogical motivations* of incorporating DVCS into CS 4900 in fall 2011. Many issues discussed here are also believed to be common concerns that many students and teachers in a CS programming course would be likely to share.

A. Student's Perspective

One of the central challenges for the students in managing their software project development is handling the *update* process among multiple distributed team members without sacrificing or introducing undue overhead. It is such a process that is too time-consuming, error-prone, and chaotic to be done either manually or using some generic Web content management tools such as Google Docs. What they truly need is an automatic version control system that has the following features –

- *Easy branching and merging*. First and foremost, every group member has a complete "sandbox" of the project. Creating branches for fixing bugs, experimenting different designs, or developing new features is easy, cheap, and fast. When the time comes to merge work outputs from multiple group members back together, even multiple times, the job is done in a snap.
- *Platform Neutral*. When multiple students work in a group nowadays like in CS 4900, it is very likely that their computers are running different kinds/versions of operating systems, mainly Windows, Mac OS X, and Linux. Therefore, they need a version control system that works seamlessly across them.
- *Distributed architecture*. Each group member can work on his/her part of the project and commit the

 ISBN: 978-1-61208-202-8

work output locally without the dependency and/or distraction of an always-on Internet connection. In the age of always-on, broadband Internet connections, we forget that sometimes we do not have access to a network [1]. This was truly a concern when literally every student in CS 4900 was working on his/her laptop; but, unfortunately, the Wi-Fi signal on VSU campus was not ideal all the time. (It was the case even on the day when students did their final project demos.)

B. *Teacher's Perspective*

As what you will see in Section V later, DVCSs satisfy all students' needs above. In fact, they have garnered significant attention in developer communities [2] while attracting relatively little in CS education. Exposing students to and familiarizing them with such an important aspect of the software development process was *the primary motivation* that drove me to include it in CS 4900. In addition, the following items were also behind the adoption of a DVCS for student projects in such a capstone class. They are elaborated in section V.

- *Preventing Plagiarization and "Free-Riders"*.
- Being unobtrusive to undergraduate level students, both conceptually and mechanically.
- *Fitting an Educational Setting*. It should not require a significant commitment of administrative, technical, and financial resources to be successful in an educational institution.

C. *Social Context*

In contrast to faculty members who belong mostly to the "baby boomers" and "X generations", college students sitting in our Computer Science classrooms today are also known as the "*Generation-Yers*" [5], who embrace mobile phones and cloud-computing based social networks as part of their daily lives. The latter is of particular importance to the subject of this paper because it includes not only general social networking sites such as *Facebook* and Web content management sites such as *Google Docs*, but also "social coding" sites such as *Github* – the primary online source code repository hosting site used by the student projects in CS4900. It is the students' digitized cultural background that makes it such a natural and smooth process to transit from the manual, tedious, and error-prone way of managing software projects to a DVCS.

III. DVCS AND GIT

Lanubile et al. did a comprehensive survey on collaboration tools for global software engineering in [2], which include Trackers, Build Tools, Modelers, Knowledge Centers, Communication Tools, Web 2.0 Apps, and of this paper's most interest, version control systems. Subversion is a popular version control system. But, it adopts a traditional centralized architecture, which does not fit well for the educational setting for reasons as described in Section II and in [2]. Git, Mercurial, and Darcs are distributed systems that operate in a peer-to-peer manner, where each local clone of the project is a full-fledged repository with complete history and full revision tracking capabilities, not depending on network access or a central server.

Although there are technical differences between these DVCSs and the decision of choosing Git in CS 4900 was quite of my personal preference, there were a few legitimate factors that reinforced my decision: first, Git is the built-in version control mechanism of Rails, the platform students used to build their Web server applications in this class; second, *Eclipse*, the Integrated Development Environment (IDE) students used for both Rails and Android programming has a Git plug-in that makes version control a natural step in their project developing cycle; and third, Github [3], the most popular online Git repository hosting site, offers educational accounts to host not only public but also private repositories for free, which is greatly convenient for authenticated accesses to both individual and group projects in CS4900.

There are some drawbacks of using Git that one needs to put under consideration. First, Windows support is still lagging behind. You simply cannot use Git from a normal command prompt. Second, there is a long and rough learning curve for students before they feel comfortable using Git.

Next, Section IV discusses how Git is used in CS 4900 from the mechanical view that focuses on various components in such a distrusted system; then Section V covers it from the "Service" view, i.e., workflows that demonstrate how the students and teacher can use and take advantages of Git.

IV. GETTING GIT TO WORK

A. *Local Git Repositories*

For each project, either individual or group, each student has a local Git repository (see the `.git/` directory in Figure 1). The *working tree* is student's current view into the repository [6]. After making changes to the files on the working tree, through the *staging area*, he/she can *commit* the changes to one of the working sets, known as *branches*, in his/her local repository and store a log message/comment explaining what the change did. (The use of such logs and comments will be more covered later in Section V.)

Each student could have as many local branches as he/she wants and *checkout* anyone at any time to start/ continue to work on it. Among these local branches, one is of special importance – The *master* branch serves as the "interface" branch to other group members and the teacher. It always stores the most current version of the project, which gets *pushed* to other students in the same group for sharing the collaborating purposes or to the teacher to be graded. When a newer version of the project from some other group members, or when a graded version of the project from the teacher, becomes available, it gets *pulled* in onto the master branch.

 ISBN: 978-1-61208-202-8

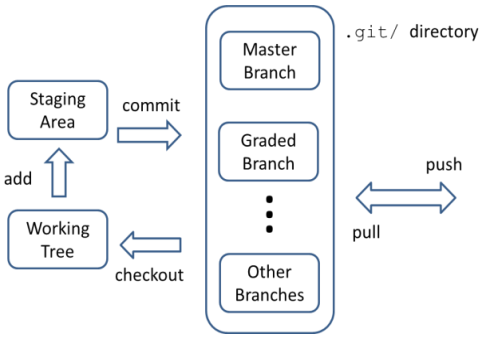

Figure 1. Student's Local Repo per Project

Teacher's project-based local repository looks structurally similar to Students', except that its branches hold graded code turned in by students, i.e., one branch per student (see Figure 2).

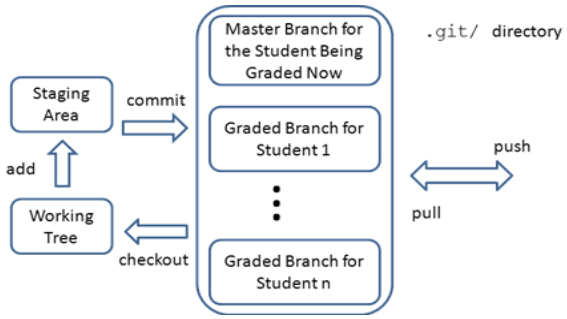

Figure 2. Teacher's Local Repo per Project

The Git push and pull operations described above are actually performed to and from online Web-based hosting service at *Github* (see Section III.B) and might incur conflicts handling and branch merging [6].

Git plug-in for Eclipse makes it very easy for all Git operations to be conducted from within Eclipse either through GUI items or more conveniently in an embedded shell (see Figure 3).

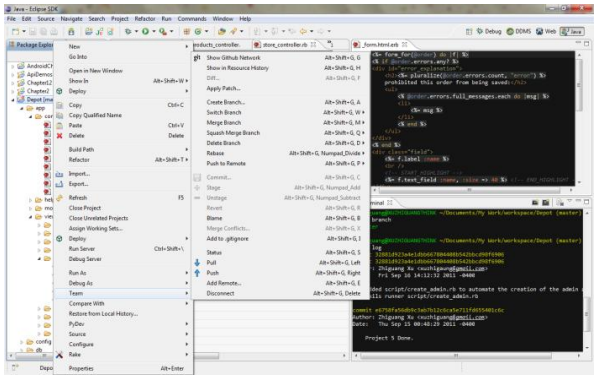

Figure 3. A snapshot of Git in Eclipse

B. *Online Public and Private Repositories on Github*

Github was chosen as the online Git repository hosting site for CS 4900 due to the following reasons:

- Free public AND private repositories, thanks to Gitbub's educational program, that allow students and teacher to access their projects from anywhere at any time
- Secure source code backup in the Cloud (True story – one student's laptop crashed in the middle of the semester and it was his backups on Github that saved his project)
- Clean and fast submission and grading of projects, especially when their sizes go beyond megabytes
- Rich tools for administrating student groups, visualizing students' contributions to their group projects, archiving projects for future course assessments, and much more

We created an *Organization* "VSU-CS4900" on Github that has 13 members (12 students and 1 teacher) and 17 private repositories (12 for individual projects, 4 for group projects, and 1 for the teacher, see Figure 4).

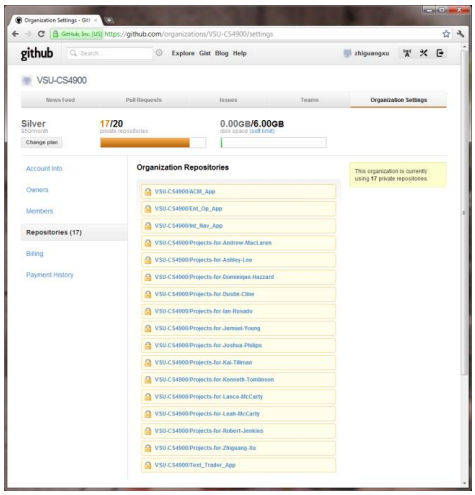

Figure 4. Private Repositories on Github

Each individual repository has two owners – a student (e.g., Ian) and the teacher – who both have full privileges, and a number of branches. The *master* branch always stores the most up-to-date version of the current individual project (e.g., #5) that Ian is working on. One the due date, the teacher will pull the project on the master branch and grade it. Once the grading is done, the graded project is pushed up to the *Project_5_Graded* branch on Ian's individual repo for him to review. There might be other branches in his individual repo that Ian creates for himself (see Figure 5).

 ISBN: 978-1-61208-202-8

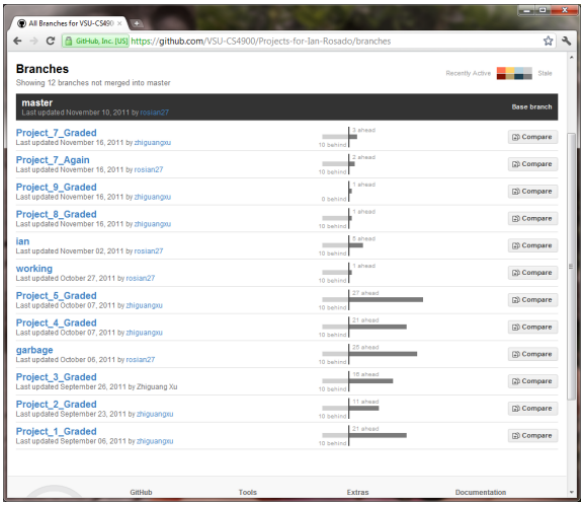

Figure 5. A Student's Individual Repo on Github

Teacher's repo has only one owner (the teacher) and a number of branches. The master branch as usual serves as the interface branch and the rest branches store the solutions to the student projects and example projects for class lectures (see Figure 6).

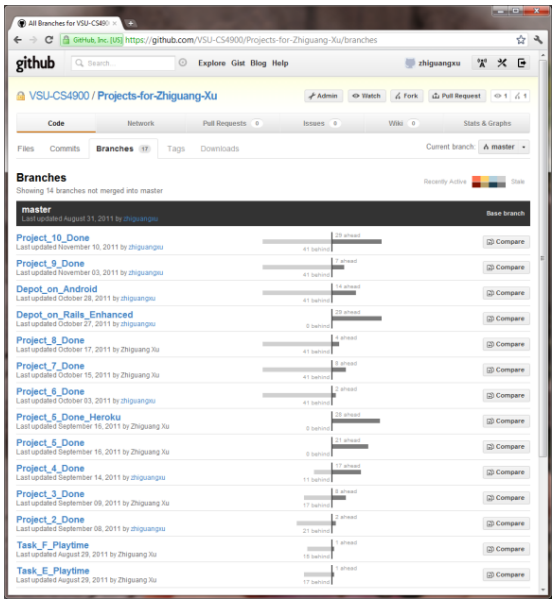

Figure 6. Teacher's Repo on Github

Each group repo has four owners (three students in the group and the teacher) and a number of branches, at least two of which store the final version of their client side code and server-side code respectively (see Figure 7).

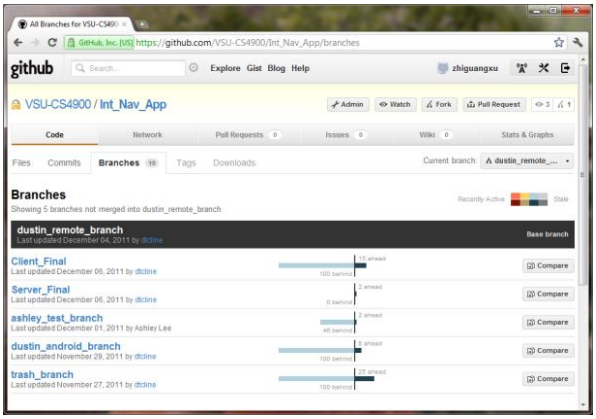

Figure 7. A Group Repo on Github

V. THE WORKFLOWS WITH GIT

This section presents three workflows with Git to show what kind of services Git (configured in the way as described in Section IV) provides to the students and the teacher to increase the overall productivity of the whole class. In the end, you will also find discussions on a challenging issue that we have encountered and how we addressed it.

- Developing, Submitting, and Deploying Projects. In a group of three students A, B, and C, student A is the "group leader" (see Figure 8). As the project progresses, each student is able to push and pull his/her newest work output to and from his/her remote branch in their group repo on Github. Only A has the privilege to pull code from everyone's remote branch, merge them, and push the result to the master branch, which consequently stores the most current version of the project for everyone to pull so as to be code-synchronized.

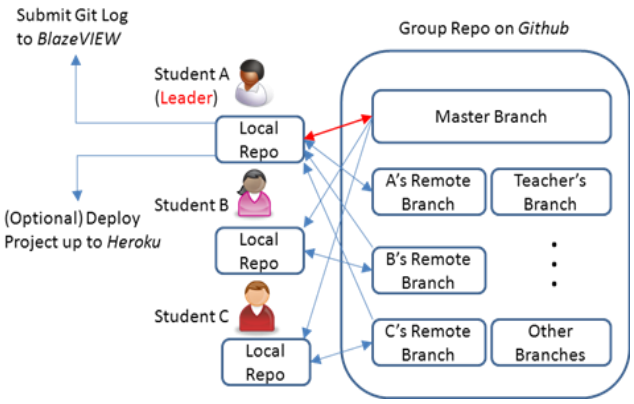

Figure 8. Student Workflow

When the project is finished, the group leader will submit the Git log file to BlazeVIEW, a Blackboard based online course management system at VSU, and optionally deploy the server side of the code to

Heroku [7], a cloud based, Rails friendly application platform.

- Grading Projects. When grading a project, the teacher pulls the code from the master branch in the project's Github repo, builds it locally, and runs it. But more importantly, the teacher heavily relies on the revision logs to see each group member's contributions to the final project (see Figure 9). These logs also provide an audit trail for determining if students followed the incremental process, which will be demonstrated by a logical and coherent sequence of commit messages that indicate a methodical progression toward the end goal. At each commit, students must stop and describe their work in a commit comment, which forces reflective pauses and helps promote an intentional attitude toward their work [4]. The graded project is then pushed up to the Teacher's branch on Github for students to review.

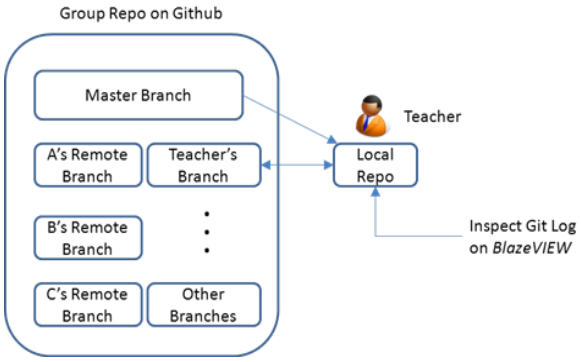

Figure 9. Teacher Workflow

In fact, the flexibility that Git extends in terms of setting up local and online repositories greatly helped how students' projects were graded in CS 4900. In addition to the semester-long projects as mentioned in the introduction section of this paper which constituted the major component in CS 4900, there were around ten "practice" projects that were designed to get students technically ready for their "big deals" (Note, Ruby and Rails and Android programming were new to most of the students in this class), and they had to accomplish these "practice" projects individually. On the other hand, a related issue that concerns lots of CS teachers (me included) is how to assess individual student's performance in group projects. Obviously, the best way to detect cheating in individual projects and free-riding in group projects is to have a version control system that comes with a rich and sane logging history that records each and every commit of intermediate work output along the evolvement of the project, based on which students can justify their progresses towards and contributions to the final product.

- Discussing Example Code in Class Lectures. For better student-teacher interactions and more efficient use of the class lecture time, Git makes it very easy for the teacher to checkout a new branch and elaborate critical code step by step to students in class (see steps 1 and 3 in Figure 10) and skip non-essential parts by checking out the commits that conclude them (see steps 2 and 4 in Figure 10) and move on.

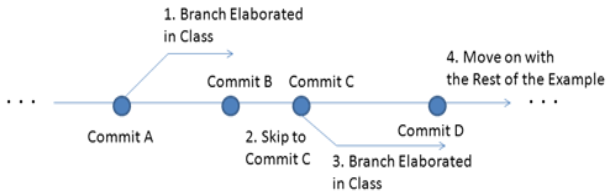

Figure 10. Discussing Examples in Class Lecture

- Challenging Issues. In addition to the bumpy road in the beginning of the semester mainly to get familiar with Git, inevitably, there were a few issues students encountered that held them from moving on with their projects but fortunately found solutions to [8]. For instance, although Git worked perfectly with Rails for the development of their servers, it gave students hard time merging work outputs from multiple group members on the Android client side into one new version by generating all sorts of conflicts. They found that the .gitignore file was their friend which allowed them to specify which files they did not want Git to track, specifically for Android, the ones in the bin/ and gen/ folders, for they will be automatically generated during the build process anyways.

VI. Conclusions and Future Work

Our initial experience with Git and DVCS in general has been very positive. We have seen senior students in the Capstone class voluntarily and comfortably use Git as the distributed version control system for their projects. Git gives them unique opportunities and exposures to collaborative and real-world practices that are prevalent in today's distributed software development community. As the students gain experience and competitive skills with the version control system that will be integrated into CS 4900, such skills scale with them, enabling them to collaborate with their peers, contribute to open source software projects, and eventually transfer their new knowledge to the work environment [2]. It also streamlines my work as a teacher in terms of grading student projects and giving lectures.

Future work includes expanding the adoption of Git in a wider range of Computer Science classes that emphasize students' programming skills. In particular, we are also going to enrich the process of composing Git commit comments [9] to help keeping them from getting too general, vague, and/or uninformative.

 ISBN: 978-1-61208-202-8

REFERENCES

[1] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaíno, "Collaboration Tools for Global Software Engineering," IEEE Software, March/April 2010, pp. 52-55.

[2] B. de Alwis and J. Sillito, "Why are software projects moving from centralized to decentralized version control systems?" CHASE '09: Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering, IEEE Computer Society, Washington, DC, USA, 2009, pp. 36–39.

[3] http://www.github.com, retrieved: June, 2012.

[4] D. Rocco and W. Lloyd, "Distributed Version Control in the Classroom," ACM SIGCSE'11, Dallas, Texas, USA, March, 2011, pp. 637–641.

[5] G. Thiruvathukal, K. Laufer, and D. Dennis, "Moving Academic Department Functions to Social Networks and Clouds: Initial Experience," IEEE Computing in Science and Engineering, September/October, 2011, pp. 84–89.

[6] T. Swicegood, "Pragmatic Version Control Using Git," Publisher: Pragmatic Bookshelf, ISBN: 1-934356-15-8.

[7] http://www.heroku.com, retrieved: June, 2012.

[8] C. Bird, P. Rigby, E. Barr, D. Hamilton, D. German, and P. Devanbu, "The Promises and Perils of Mining Git," MSR '09 Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories, Washington, DC, USA, 2009.

[9] M. D'Ambros, "Commit 2.0: Enriching Commit Comments with Visualization," ICSE '10, Cape Town, South Africa, May, 2010.

 ISBN: 978-1-61208-202-8