Revision Control System (RCS) in computational sciences and engineering curriculum

Gowtham S*
Michigan Technological University
1400 Townsend Drive
Houghton, MI USA
g@mtu.edu

ABSTRACT

The Revision Control System (RCS) is an essential aspect of software development process and software configuration management. While continuing to be an integral component of the *real world*, it is often left out of the main stream curriculum in most academic institutions. Instead, students are expected to learn it on their own, as a hobby or as an independent study, out of personal interest. The author describes the experiences gained from attempting to implement a distributed Revision Control System, Git, as part of the computational sciences and engineering curriculum at the undergraduate and graduate levels. The author also describes the advantages for both parties involved: improving the competency of students and preparing them for the real world expectations while providing the teacher an opportunity to provide timely feedback to the students and monitor their progress. The availability of free and open source tools used to analyze and visualize the commit history to the repository helps teachers and students observe submission and feedback patterns respectively.

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education—*Collaborative Learning, Computer-Assisted Learning*; K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Curriculum, Information Systems Education*

General Terms

Design, Experimentation, Management, Reliability

*Dr. Gowtham is the Director of Research Computing in Information Technology Services, and an adjunct assistant professor in Physics, and Electrical and Computer Engineering.

XSEDE '14 July 13 - 18 2014, Atlanta, GA, USA

http://dx.doi.org/10.1145/2616498.2616576.

Keywords

Computing, Computational Sciences and Engineering, Education, Git, GitHub, GitLab, Gource, Programming, Programming Etiquette, Revision Control System, Teaching

1. INTRODUCTION

There has always been a need for a logical and consistent way to organize and track the revisions of documents of all kinds. The most commonly used approach to organize but not necessarily to track the changes is to keep multiple physical copies of a given document – tagged in some convenient way to uniquely identify a revision. This often takes one of the following forms, for example, for a text document: `filename.txt`, `filename_1.txt`, `filename_2.txt`, `filename.txt.bak`, `filename.txt.MMDD`, `filename.txt.YYYYMMDD` and so on. The lack of a simple way to revert back and forth between such revisions, and the lack of ease with which such revisions can be compared makes it difficult to confidently rely on this kind of an approach.

Revision Control System comes in two kinds: centralized and distributed. Concurrent Versions System[1] (CVS) and Subversion[2] (SVN) are examples of the centralized flavor while Bazaar[3], Git[4] and Mercurial[5] are examples of the distributed kind.

In an effort to improve the competency of students, help them be better prepared for the expectations of the real computing environments, ensure that the students do not repeat the author's own mistakes, and do so via *learning by doing* instructional strategy, the Revision Control System was introduced into the computational sciences and engineering curriculum for undergraduate and graduate students in Michigan Technological University in the Fall of 2010.

Details of the curriculum set up, instructional strategies and lessons learned from the process are described in the following sections.

2. CURRICULUM SET UP

Even though the syllabus initially used Subversion, Git, a popular distributed Revision Control System became the

[1]http://www.nongnu.org/cvs/
[2]http://subversion.apache.org
[3]http://bazaar.canonical.com
[4]http://git-scm.com
[5]http://mercurial.selenic.com

adopted methodology since Spring of 2013. GitHub[6], the largest open source community in the world, smoothens the learning curve by providing an online platform and handful of graphical utilities for all operating systems. To protect the privacy of students' submissions and safeguard against plagiarism, the author chose to use GitHub's private repositories at a minimal cost.

Computational Methods (PH4390; Fall) and Computer Simulations (PH4395; Spring) were the two courses in which Git was implemented. While these were listed as undergraduate courses, several graduate students enrolled every time the courses were offered. The combined objectives for these courses are listed below:

1. Gain a first hand experience of the use and limitations of computers for solving problems in physical and mathematical sciences and in engineering.
2. Acquire/Enhance good programming and communication etiquette.
3. Translate problems in physical and mathematical sciences and in engineering into computer programs (C, C++, FORTRAN, Mathematica, MATLAB, Python, etc.) to be solved using computational routines and resources.
4. Understand and troubleshoot the (sources of) errors in scientific and engineering programs.
5. Learn the nuances of scientific computing and visualization.
6. Incorporate others' scripts, tools, modules and routines into one's own workflow and vice versa.
7. Gain hands-on experience in data center etiquette, systems administration, security and compliance by designing and managing a linux HPC cluster for research computing.

An identical set up was used for both courses: each student was granted access to her/his own repository owned by the teacher. The in-class lectures and demonstrations as well as the course material established parts of the repository as read-only and read-write for both the student and the teacher. This was necessary to ensure the content created by the teacher wasn't inadvertently over-written and/or deleted by the student, and vice-versa.

3. INSTRUCTIONAL STRATEGIES

1. Majority of the students, 50+%, were not aware of either the concept or the practice of a Revision Control System. This was true every single term the course was offered. Such information was gathered via a pre-course survey with following questions:
   (a) What is your operating system of choice?
   (b) Do you know any programming language? If yes, assign a numerical score to each language on a scale of 1 to 10 (1 being *I just know the name of it* and 10 being *I pretty much know everything there is to know about it*).
   (c) Where do you see yourself at the end of this course? In other words, what is your expectation from this course? Are there specific goals you wish to accomplish in this course or as a result of completing this course?
   (d) Have you heard of or used a Revision Control System?
2. In an attempt to encourage students to start consistently working on the assignment from the day it is handed out, no help was offered during the 48 hours before the deadline.
3. To encourage students to cultivate the habit of using Revision Control System, it was made the only available method for distributing course materials and accepting assignments/projects. Email attachments and hard copies were deemed unacceptable submission formats.
4. To further encourage the students to accomplish #2 and #3, each assignment was designed with a clear expectation that there had to be several intermediate submissions. Similarly, the term project was designed with weekly status reports.
5. Finally, students were informed that adopting the Revision Control System would not yield instant gratification, and that the impact would be observable by making a conscious effort to practice it over a considerable length of time.

4. LESSONS LEARNED

1. Adopting Git gave the teacher an opportunity to distribute the course materials to students with relative ease, and provide timely feedback to students' (partial) submissions.

   Students were given the opportunity to turn in partially completed assignments and seek teacher's feedback, if need be, before the final due date.

   In either case, the chance of missing a deadline due to lost hard copies, and forgetting to attach the documents in an email were significantly minimized.
2. Given the online nature of GitHub and its accessibility from just about anywhere, it provided the students an opportunity to continue working from different locations without missing a beat.
3. Using Git also served as another form of data back up so that teachers and students alike did not lose semester-long work due to failed hard drives and/or computers.
4. Availability of free and open source tools to visualize the commit history such as Gource[7] provided teachers, students and the administrators an opportunity to observe the evolution of an assignment and/or a project over a period of time.

[6]http://github.com

[7]https://code.google.com/p/gource/

5. Not every student bought into the concept and practice of using a formal Revision Control System. The majority of such students seemed to understand the concept but showed an aversion to computing, programming and/or learning newer technological topics.

6. Students who consciously bought into the concept and practice of using Git were generally the ones that had an interest in learning newer technological topics.

   With timely (partial) submissions of assignments and status reports, their interaction time with the instructor increased significantly. They made use of this opportunity, included the suggested corrections to revise their submissions and in turn, developed good, competent work flows to write programs, and kept their documents comparatively better organized.

With graduated students providing feedback to the department and to the teacher regarding the applicability and usefulness of a Revision Control System in their real world positions (universities, research institutes and national labs, industries in and outside the US), there has been a justification for its formal inclusion in the curriculum.

5. FUTURE WORK

With the relative success of PH4390 and PH4395, efforts to evolve them into a university-wide curriculum at the 5000 level (UN5390: Scientific Computing I and UN5395: Scientific Computing II) and make them an integral part of the upcoming Data Sciences (MS and graduate certificate) program have been successful.

The undergraduate Physics curriculum is also being revised to incorporate various aspects of computing at the sophomore and junior levels.

Starting in the summer of 2014, Michigan Technological University will be using GitLab[8], an internally hosted platform akin to GitHub, and will offer it to the entire campus community as a service from the Information Technology Services. Experiences discussed so far have been a driving force in designing and developing usage policies, procedures, and APIs that automate a variety of tasks.

6. ACKNOWLEDGMENTS

The author is grateful to GitHub for offering an educational discount[9] since February 2014. The author would like to thank Dr. Walter Milligan (CIO) and Daniel de Beaubien (CTO) from Michigan Tech for accommodating a flexible work schedule that allows teaching; Dr. Ravindra Pandey and the Physics department in Michigan Tech for not only the opportunity to teach two courses but also for the freedom to incorporate newer topics; Dr. Steven Gordon from the Ohio Supercomputer Center, Dr. Rubin Landau from the Oregon State University, Dr. Henry Neeman from the University of Oklahoma, Dr. Gregory Odegard and Dr. Warren Perger from Michigan Tech, and members of the local Houghton-Hancock community for constant encouragement, having an open-door policy to brainstorm ideas, and for sharing their invaluable experience.

[8]https://www.gitlab.com
[9]https://education.github.com