Subverting the Fundamentals Sequence: Using Version Control to Enhance Course Management

Curtis Clifton, Lisa C. Kaczmarczyk, and Michael Mrozek
Department of Computer Science and Software Engineering
Rose-Hulman Institute of Technology
5500 Wabash Ave.
Terre Haute, Indiana 47803-3999
{clifton, kaczmarc, mrozekma}@rose-hulman.edu

ABSTRACT

Instructors of introductory courses face many challenges, not the least of which is dealing with a large volume of course materials and students with differing backgrounds. There are often too many administrative demands to have as much time for creative pedagogy as one would like. Team projects, and complex realistic projects in general, increase psychic demands, and conflicting schedules make creative collaboration with other instructors impossible. In order to address these issues, we need to find ways to increase effective handling of course development, to free up time for creative pedagogical efforts. This paper reports on an exploratory project in which two instructors and an undergraduate teaching assistant used the Subversion version control system to collaborate remotely on developing and running two CS1 classes. We focus on the ease and efficiency of course management using Subversion, providing a new perspective on how version control can enhance teaching.

Categories and Subject Descriptors

K.3.1 [**Computer Uses in Education**]: Computer-managed instruction; K.3.2 [**Computer and Information Science Education**]: Computer science education

General Terms

Management

Keywords

Version control, Subversion, CS1, fundamentals sequence, course management

1. INTRODUCTION

Complex projects, and working in teams, are often desirable in early CS courses in order to provide motivating projects and realistic applications. To increase retention it is important to engage students' interests and provide a supportive environment at the beginning of the curriculum. However, it is hard to manage complex projects in introductory courses. For example, in many institutions, CS1 is a large lecture class, with an unwieldy volume of instructional material to manage. Time management problems are exacerbated by the addition of complex projects. Instructors of different sections may want to collaborate, but such work may not venture beyond sharing of a syllabus, assignments and exams. Conflicting class schedules and the unique needs of beginning students make it hard to find time to creatively collaborate on course development and management. In order to avoid being overwhelmed by administrative tasks, instructors may feel they do not have time to focus on creative pedagogy with complex group work in CS1.

Industry deals with software complexity and collaborative development efforts using version control systems. Given the similarity of the administrative challenges faced by several instructors managing a large group of students and a team of software developers working on a distributed project, it makes sense to find a way to enlist version control in CS1.

The primary goal of this paper is share the results of using version control in CS1 to reduce administrative demands and to support creative collaboration between two instructors and an undergraduate teaching assistant. This paper presents the results of an exploratory project in which the Subversion version control system was integrated into two sections of a CS1 class that uses complex Java pair and team projects. The three members of the instructional staff rarely met in person, yet interaction was highly effective between them. Version control reduced duplication of development effort, and simplified organization of shared materials. Because students also used version control, instructors were able to keep tabs on student progress easily. This monitoring often led to spontaneous course adjustments during the term – without lengthy in-person meetings. The undergraduate teaching assistant used version control to provide timely feedback to students. All of these benefits of using version control allowed the instructors to focus their energies on their primary mission: the pedagogical demands of providing CS1 students a successful learning experience.

2. BACKGROUND

2.1 What is Version Control?

A *version control system* is a set of tools for managing changes in a file or set of files over time. A typical version

SIGCSE'07, March 7–10, 2007, Covington, Kentucky, USA.
Copyright 2007 ACM 1-59593-361-1/07/0003 ...$5.00.

control system includes a central *repository* containing the current version of all files. The central repository also stores documentation of all changes made to the files along with data for restoring any past versions. Each user of the system typically keeps a working copy (or several) on his or her computer. Popular version control systems include the Concurrent Versions System (CVS) [2] and Subversion [3].

Many professional software developers consider version control to be an essential part of the software development toolkit [9, §28.2][11, §3]. Hunt and Thomas point out that version control is like a global "undo" operation that works over days and weeks, not just over the current editing session [5, §17]. Software developers can also use version control as an aid in debugging, by performing a binary search over the versions since the last known good one.

2.2 Previous Applications of Version Control

Version control systems are beginning to be integrated into coursework at different levels of the curriculum. Most previous work with version control in the CS classroom has focused on how it can provide more realistic software development experiences for students, but none has tackled course management issues in a CS1 course. In addition, most previous work has used CVS, however a more flexible alternative is now available.

Hartness [4] proposed having students use CVS for software development. Linder et al. [7] had students use CVS in a project-intensive third course. Although acknowledging the critical importance of using version control for complex projects, this very interesting paper provides few details on how CVS may have been used on a day to day basis by instructors or students. Liu et al. [8] used CVS to visualize team and individual contribution in a software engineering class. Their study demonstrates that repository data contains a wealth of information the instructor can use to monitor team and student progress. As in the preceding paper, this work does not discuss any application of version control for instructional development and course management.

Reid and Wilson [10] used CVS to provide students a more realistic software development experience in a CS2 course. Interestingly, the authors also describe how CVS affected their efficiency and workload, using this opportunity to point out several important drawbacks to using CVS. These problems included high maintenance, high storage requirements for large classes, students accidentally damaging repositories, and security issues including the potential for student cheating. However, they noted that these problems were not inherent to version control and could be avoided or eliminated by using another system such as Subversion.

Subversion has many advantages over CVS, while remaining open source. Unlike CVS, Subversion allows users to modify directory structures and rename files while retaining historic information. This feature allows the instructor to track all student work rather than potentially losing some information. Subversion is also able to perform more operations without connecting to the central repository than CVS, thus reducing the time before the instructor or student can continue his or her work. Unlike CVS, which uses Unix file permissions for access control, Subversion can use a single text file to describe fine-grained access control on a per-team or per-user basis. The greatly reduces the effort required to maintain a secure system. Subversion makes more efficient use of server disk space than CVS does, making it a better choice in resource-constrained environments. Finally, students only interact with their repositories via the Subversion tools, avoiding the damaged repositories that Reid and Wilson saw with CVS [10].

There has also been some work on using version control for managing non-code artifacts. For example, an interesting case for using version control to manage complex intellectual work comes from an unexpected setting: collaborative creative writing. Lee et al. [6] lay out a detailed argument for using version control to reduce workload and help manage document development between several busy or geographically remote people. They make a compelling argument for not only using version control to assist in collaborative writing, but for doing so asynchronously. The benefits of using version control for team document development are thoroughly discussed, but unfortunately the authors have not yet implemented their ideas.

The prior work, and the improvements that Subversion provides over CVS, clearly indicate that there is an opportunity to use Subversion to support collaborative course management.

3. IMPLEMENTING THE PROJECT

3.1 People and Their Needs

The Rose-Hulman Institute of Technology (RHIT) is a small, private science and engineering college operating with three 10-week terms. Course work proceeds rapidly, with students in all courses expected to complete substantial design and implementation. The 12 Computer Science and Software Engineering (CSSE) faculty frequently collaborate on course development, sharing instructional resources such as tests, assignments, and sample programs. This is especially true for the introductory classes (CS1, CS2), which are commonly taught by several different instructors each year, often during the same term.

Currently, CS1 uses pair programming [1] for several complex OOP assignments and teams of 2-4 students collaborate on two large projects. Managing these constantly changing teams can mean time consuming behind the scenes work. As at many other institutions, CS1 and CS2 are taken by potential majors as well as students filling a requirement for another degree. Therefore prior experience and motivation vary widely. Instructors need to reduce the time spent on administrative tasks so that they can concentrate on pedagogical issues.

To address these administrative needs, and to support collaborative course management, the instructional staff began experimenting in the fall of 2005 with using version control as a course management tool. Subversion was chosen because it is open source, and (as discussed above) has several advantages over CVS. In this first phase, five course faculty used Subversion to store and update their personal instructional materials. Faculty shared some material using version control at this time, but version control was not used to manage assignments.

The second phase, and the focus of this paper, was to use version control to collaboratively develop and update all course materials during a term. This meant also using version control to distribute, collect, and evaluate all student programming projects across multiple sections of a course. Two unusually small sections of CS1 provided a low risk opportunity to evaluate this new application of version control. In addition, one of the CS1 instructors had extensive experience with version control systems, while the other instructor did not. The teaching assistant (an undergraduate, and third author of this paper) had no experience using version control. This diversity of experience, along with the small class sizes, provided a unique opportunity to evaluate the use of version control in the complex environment of CS1. The goal of the exploratory study was to find out if version control could be successfully used in a CS1 course to enhance the efficiency and effectiveness of collaborative course management, so that more time and energy would remain for actual teaching.

3.2 Version Control Tools Used

As noted in the previous section, the CSSE department uses the open-source Subversion version control system [3]. The central repository is hosted on a Linux-based server. The introductory sequence uses the Eclipse integrated development environment. Students and instructors use the Subclipse plug-in for Eclipse to manage working copies of Java source code. TortoiseSVN is used for non-source-code files.[1]

The tools used in the course were the same tools used by the instructors for course management in this study. The instructor with extensive experience served as primary administrator of the Subversion server.

3.3 Managing Assignments and Projects With Version Control

Version control facilitated all phases of every student project in the course. Figure 1 shows the phases of a typical project, starting at the upper left. The rectangles represent instructor actions. The instructors would distribute template code to all students by running a couple of simple scripts on the department server. (These scripts are available from `http://www.rose-hulman.edu/~clifton/svnScripts` or by contacting the authors.) After distributing the template code, the instructor would demonstrate a finished project for his or her class: a single Subclipse operation sufficed to checkout the finished code from a secure section of the repository.

After the class demonstration, students would begin working on the project. Parallelograms in Figure 1 represent student actions. Students worked in parallel, regularly sharing code and other artifacts with teammates using Subclipse and TortoiseSVN. Though not shown in the figure, the instructors could monitor student work during this time by using a single Subversion command to grab a current snapshot of their progress.

Because all work was stored in the central repository, students did not have to explicitly turn in work at the due date. Instead, the teaching assistant (see ellipse in Figure 1) would use a single Subversion command to retrieve copies of all the student work. He would assess the work and add feedback directly within it. Another single Subversion command would distribute the feedback to every student on the team.

Finally, each student would use a single Subversion command to get an updated version of their work that included the assistant's feedback.

4. EVALUATION OF VERSION CONTROL FOR COURSE MANAGEMENT

4.1 The Instructors' Perspective

Subversion reduced instructor workload and improved time management in several ways. For example, the instructors had only two free hours a week in common, which meant that those meetings had to give priority to topics needing in-depth discussion. Subversion allowed these face-to-face meetings to be highly productive, because they excluded the more mundane tasks on which the instructors worked asynchronously but collaboratively. These tasks included the development of schedules, course documents, and paperwork required for accreditation.

4.1.1 *Distributing Materials*

The instructors also used the version control system for maintaining the course web site. The main course repository included a "Public" subdirectory that contained the entire course web site. It was easy to develop or revise materials by just editing local copies. Often one instructor would be updating homework assignments while another updated presentation materials. This work proceeded in parallel. When the materials were ready for publication, a single Subversion command updated the working copy of the Public subdirectory on the web server. It took mere seconds to combine and publish the materials for class use.

Version control also let the instructors quickly and easily create and distribute new sample exercises. On one occasion, the instructors decided that an additional interactive example would help to solidify a concept. After creation, the complete distribution of the exercise only took three minutes and did not require writing any instructions to the students about how to retrieve it.

4.1.2 *Interactions with Students*

Version control simplified dealing with student absences. For example, on one occasion a student was ill and missed several days of class, by which time the other students on his team had started and completed a project. It took the instructor about 30 seconds to created a new repository for the student and give him a clean copy of the template so that he could complete the work individually. The version control system then allowed the assistant to grade this student's work without any additional instructor intervention. On another occasion a student missed the last day of a pair programming assignment. His partner had already completed the assignment when the first student returned to class. The instructor was able to create a new repository for the absent student that contained the last collaborative work of the pair. The student could then complete the project on his own, demonstrating his competence with the material. Again, the instructor effort to do this was minimal.

Version control also reduced administrative load by allowing instructors to provide direct and timely feedback to students. When grading was complete, the instructor or assistant packaged the results, along with a list of comments and suggestions, and committed the entire compilation to the students' repositories. It was not necessary to wait for class to meet to hand back paper copies. The students received feedback sooner than under previous assessment methods, and in a neat predictable format. In addition, the repository maintained an easily accessible record of progress.

[1] Subversion server software, Subclipse, and TortoiseSVN are all available from `http://scm.tigris.org`. Eclipse is available from `http://www.eclipse.org`.

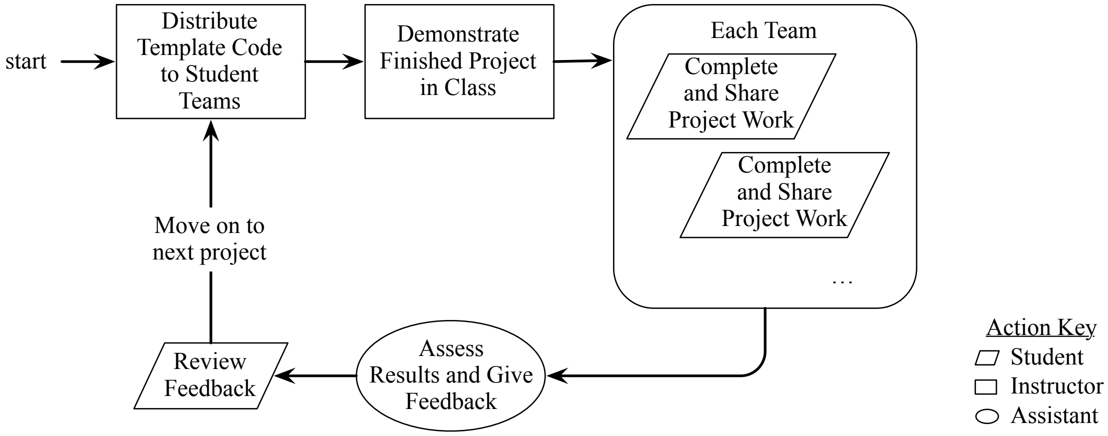
start
Distribute Template Code to Student Teams
Demonstrate Finished Project in Class
Each Team
Complete and Share Project Work
Complete and Share Project Work
...
Assess Results and Give Feedback
Review Feedback
Move on to next project
Action Key
▱ Student
□ Instructor
○ Assistant

Figure 1: Version control at all phases of a student project in CS1.

Even before official grading began, version control made it very easy to do "spot checking" on student work; at any time, without having to ask the students to turn in anything, the instructors could look at the student repositories to find out if projects were being checked in, what changes were being made and who was making the changes. Thus the instructor could arrive at class prepared to target individual students or teams who appeared to be having difficulties.

This ability to look in at work in progress also streamlined the handling of student questions. When a student asked for help during office hours or by email, the instructor could quickly checkout a current version of the student's project. This let the instructor focus on helping the student understand and solve his or her problem. Email requests for help that previously would have resulted in a lengthy series of messages were often dealt with in a single exchange.

4.2 The Assistant's Perspective

RHIT uses its undergraduate assistants primarily for grading, and so the teaching assistant offers a unique perspective on the benefits of version control for assessment. As mentioned previously, our CSSE courses emphasize the value of teamwork. While our projects are constructed to encourage team-oriented work, a risk is that sometimes a minority of students do a disproportionate amount of the work. This is a particular concern in CS1, where we want to ensure that each student gets off on the right foot. Without version control, trying to determine which team member had done which part of a project was at best difficult, and at worst, impossible. The longer a project took, the less distinct each student's contribution would become over time.

When we used version control, this problem was subverted. This is because one of the features of version control systems is that they accurately and automatically provide (after the fact) both a task list and an iterative enhancement plan via the revision log. (Liu et al. [8] pursued a similar idea.) While students may not have fully appreciated this feature, we could use it to determine who changed what, and when. Apropos, the very mechanism that encouraged teamwork on a project also enforced it.

As an example of how effective this method of evaluation can be, we examined repository data collected for the final project of the Spring CS1 course described in this paper. In one case, a pair of students with 46 revisions split them exactly, with each partner committing 23 changes, showing excellent division of tasks. In another case, despite the fact that the students on a team forgot to assign tasks on their original iterative enhancement plan, the revision log clearly showed that one student made only 18 of the nearly 200 project revisions. On a three person team, responsible for 138 revisions, two of the students split the work, making 54 and 44 revisions each, while the third made only three revisions. By breaking the data down by weeks, it became clear that these three revisions happened at the end of the project; from this data we can infer that this student was busy with other things early on, or that his partners finally objected to his lack of contributions, and forced him to finish the project. These examples highlight just a few ways in which version control makes it easier for assistants (and instructors) to gain insight into student contributions to complex projects.

5. DISCUSSION AND FUTURE WORK

The small class sizes permitted a safe exploration of how Subversion could reduce administrative load and enhance collaborative course management in CS1. Following the success of this exploratory study, the next step is to apply these same tools and techniques in a larger class. It will be important to verify that the successful collaboration and live course management reported in this paper will scale up to a larger student body. There is every reason to believe these successful results will scale up, because the nature of version control is that it scales well to large projects.

From a user perspective, both the instructor with less version control experience and the assistant with no prior experience found the system very easy to learn and use. A few caveats are worth noting however. First, it was extremely helpful to have an experienced version control user as one of the faculty, as he was able to rapidly address the rare but occasional technical needs that arose. For example, navigating the features of the Subclipse plug-in for Eclipse was at first confusing to the other instructor. It was valuable to have the experienced faculty member available to provide an overview. Second, the current version of Subversion assumes that the primary administrator is comfortable in a command-line Unix environment. At this time there is no GUI front-end for managing repositories and access permissions. This could be problematic to administrators new to Unix.

Anecdotal evidence suggests that students enjoyed using Subversion as well. For example, during one class, a student experienced hardware failure moments after committing his contribution to a team project. His initial shock at seeing the "blue screen" disappeared when he realized his work was not lost and his team pumped their arms in the air and cheered "Go, Tortoise!"[2] It was also interesting to watch students intently studying their assessment notes in the repository; not one student in either section of CS1 said they did not understand their grades – by being able to easily view their annotated work, they were able to answer their own questions. Thus the instructor was able to spend more time helping with new assignments than before the use of version control.

A next step in evaluating student use of version control would be to formally study student perceptions of its value to them both as a project development tool and as a feedback mechanism. As just one example, the commit process was intuitive to the instructors but not to many of the students. It would be interesting to analyze why students found this concept difficult to grasp. Writing log messages is another fertile topic to investigate: some students wrote very useful messages, while others wrote terrible messages. It would be interesting to explore how students view the log in relation to their development efforts, and leverage that information to improve their learning.

Yet another way to investigate student learning would be to follow the model taken by Zimmermann et al. [12]. They developed a system that analyzed version control log files to determine which files changed simultaneously. Based on this analysis, the system could suggest likely oversights when users later changed just one of a set of (probably) mutual dependent files.

There are many interesting pedagogical questions that could be answered by conducting a thorough statistical analysis of the log files. For example, the number of commits per student is only an imperfect predictor of the amount of the work done. One student might work an entire afternoon on a project and commit once at the end. Another might be afraid of losing data and so commit after every change. Beyond the obvious fact that numbers alone do not tell everything about what a student is doing, what might these data reveal about the working styles of different students on a team? It would be interesting to investigate how this information could be used to enhance individual and team learning.

6. CONCLUSIONS

The results shared in this paper provide a new perspective on how version control can be used in CS1. In particular, they show how Subversion can be used to support instructor collaboration and creativity, by reducing administrative demands, increasing effective communication, and streamlining interaction with students. The central repository and running logs enabled two instructors and an undergraduate teaching assistant to develop and dynamically manage a high volume of instructional materials and student deliverables effectively. As a result, the instructional staff had more free time to concentrate on pedagogical issues in an introductory class that used complex team projects. Many instructors collaborate on introductory classes and have a desire to implement complex projects in those classes. In Subversion they now have a ready tool to help them find the time and creative energy to do so.

7. ACKNOWLEDGEMENTS

The authors would like to thank Gary Leavens for his helpful comments; Matt Boutell, David Mutchler, and Steve Chenoweth for previous collaboration on more limited applications of version control; and Mike McLeish for initial configuration of the Subversion server used in the study.

References

[1] K. Beck. *Extreme Programming Explained.* Addison-Wesley, 1999.

[2] P. Cederqvist. CVS—concurrent versions system. Available from `http://ximbiot.com/cvs/manual/cvs-1.11.22/cvs.html`, 2006.

[3] B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato. *Version Control with Subversion.* O'Reilly, 2004.

[4] K. T. Hartness. Eclipse and CVS for group projects. In *CCSC'06.* Consortium for Computing Sciences in Colleges, 2006.

[5] A. Hunt and D. Thomas. *The Pragmatic Programmer.* Addison Wesley Longman, Inc., 1999.

[6] B. G. Lee, K. H. Chang, and N. H. Narayanan. An integrated approach to version control management in computer supported collaborative writing. In *Proc. of the 36th annual Southeast regional conference*, pages 34–43. ACM Press, 1998.

[7] S. P. Linder, D. Abbott, and M. J. Fromberger. An instructional scaffolding approach to teaching software design. In *CCSC'06.* Consortium for Computing Sciences in Colleges, 2006.

[8] Y. Liu, E. Stroulia, K. Wong, and D. German. Using CVS historical information to understand how students develop software. In *MSR 2004: International Workshop on Mining Software Repositories*, 2004.

[9] S. C. McConnell. *Code Complete.* Microsoft Press, 2004.

[10] K. L. Reid and G. V. Wilson. Learning by doing: Introducing version control as a way to manage student assignments. In *SIGCSE'05*, pages 272–276. ACM Press, 2005.

[11] J. Spolsky. *Joel on Software.* Springer-Verlag, New York, NY, 2004.

[12] T. Zimmermann, V. Dallmeier, K. Halachev, and A. Zeller. eRose: Guiding programmers in eclipse. In *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 186–187. ACM Press, 2005.

[2] Recall TortoiseSVN is one of the version control tools used.