Learning by Doing: Introducing Version Control as a Way to Manage Student Assignments

Karen L. Reid
Dept. of Computer Science
University of Toronto
Toronto, Ontario
reid@cs.utoronto.ca

Gregory V. Wilson
Dept. of Computer Science
University of Toronto
Toronto, Ontario
gvwilson@cs.utoronto.ca

ABSTRACT

Professional software developers use version control systems to coordinate their work, and to provide an unwindable history of their project's evolution. In contrast, students in most programming courses use a homegrown electronic submission program to submit their work, and email to coordinate with partners when doing team projects. In May 2003, we began using CVS, a popular open source version control system, as an assignment submission system. Students receive starter code by checking out the assignment, use the version control system to manage their work, and submit their assignment by committing it to CVS. Teaching assistants grade assignments by checking out each student's repository, and committing the marks. Our experience to date shows that this is both a simpler and a more flexible way to manage student assignments, and also an excellent way to teach them how to use a fundamental software development tool.

Categories and Subject Descriptors

K.3 [**Computing Milieux**]: Computers and Education; D.2.6 [**Software**]: Software Engineering, Programming Environments

General Terms

Education, Management

Keywords

Education, Version control, Software tools, Software Engineering

1. INTRODUCTION

Version control repositories lie at the heart of most modern commercial and open source software projects, as they allow multiple developers to share files in a safe, controlled way, while automatically creating a history of the project's evolution. As several authors have observed [5, 7, 10], one way to predict whether a software project will succeed or not is to ask, "Are developers using a version control system to coordinate their work?"

Prior to May 2003, undergraduates in Computer Science at the University of Toronto were not introduced to version control until their final-year software engineering course. As a result, most students completed their degree with only a hazy understanding of version control's importance and how best to use it. When we introduced a new core second-year course, "CSC207: Software Design", we decided to structure it so that students would acquire such understanding and skills early in their academic career.

Coming into CSC207, students had completed a traditional first-year course using Java. In that course, they used a command-line or web interface to submit their assignments, which copied their files to a course submission directory. If a student ran the submission process a second time, the original submission was overwritten.

Like most open source projects, we chose to use CVS (the Concurrent Versions System) [1]. Its main advantages are that it is free, well-documented, and available on Microsoft Windows, all major flavors of Unix, and Mac OS. Students had no difficulty installing and running it on their personal machines.

In order to ensure that students mastered CVS, we required them to use it to submit their assignments, rather than a traditional electronic submission system. As well as introducing students to a core professional tool, we believed this would:

- make it easier for students to work in pairs or larger groups (an essential skill that is too often shortchanged in undergraduate programs because of the administrative overhead of setting up groups);
- make it easier for students to work on both personal machines and university lab machines;
- make it easier for instructors and teaching assistants (TAs) to assist students during office hours, since they could get an up-to-date copy of students' work at any time; and
- provide a history of student work, which could both help us identify patterns in the way they programmed, and give us additional information in cases of plagiarism.

SIGCSE'05, February 23–27, 2005, St. Louis, Missouri, USA.

Below, we describe why we felt it important to introduce students to version control this early, how we actually implemented CVS-based assignments, what our experiences were, and how well we met our goals.

2. PEDAGOGIC VALUE

First and foremost, introducing students to CVS early in their careers helps them see that software development is a process that can be learned and improved, rather than a "black art" which magically yields running programs. It also gives instructors an opportunity to show that good tools and working practices can be used with any language, on any operating system, for any kind of work.

Second, students (like everyone else) are often reluctant to invest time today that will only pay off in the long run. Requiring them to use CVS forces them to get over the initial learning curve. Once they do, and see how much more productive this tool makes them, they are more willing to to learn other tools and practices. This is particularly true when they recognize the advantage of integrated tools in a system such as Eclipse [2]. It also helps that our insistence on CVS forces us to lead by example. Students are more likely to believe that these tools are important when they see their instructors using them on a daily basis.

CSC207 is many students' first encounter with the differences between "toy" programs and larger software projects. While their earlier assignments consist of a handful of Java source files, using CVS allows us to introduce students to the practice of recording how their work has changed over time. Similarly, while they are taught to write meaningful comments describing the state of their code at a particular instant, writing CVS log messages teaches them to describe how that code is evolving. Their practical work therefore helps to introduce them to the concepts associated with "programming in the large".

Finally, giving students a chance to use CVS on their own for several assignments before they use it with a team ensures that they only have to deal with one set of issues at a time rather than trying master a new tool and learning to work with other students simultaneously. Separating these two issues means that when students encounter their first team assignment they will use version control properly rather than using it primarily as a means of submitting their final assignment.

3. IMPLEMENTATION

At the beginning of the term, the instructor creates a repository for each student, with a directory or module for each assignment. The instructor adds starter code or other material to each student's repository using normal CVS commands. We have written a number of simple shell scripts to automate creating repositories and adding files to them. This kind of automation is important because of the course's size: we are currently running 4 sections across 3 campuses with 3 instructors, and a total of 400 students.

It is important that students understand the underlying model, so we spend class or tutorial time explaining how to use CVS. In CSC207, we also spend class time explaining the general principles of version control, and describe how it can be used effectively by professional programmers. Students are given a web page tutorial that has specific instructions about getting started with CVS using our lab machines. One hour of lecture and one hour of tutorial seems to be sufficient to get the students started. Students run "cvs checkout" or "cvs update" to obtain starter code or other files provided by the instructor. To submit an assignment, students simply run "cvs commit" before the due date.

To grade the assignments, the teaching assistants (TAs) check out a copy of each student's repository and compile and test the programs. In the first two terms of using CVS, the TAs added and committed a marks file to each repository containing feedback and evaluation. This led to some confusion since it was not clear to students when the marking had been completed, and it was possible that students saw partial marking results. At present, the TAs edit the marks files in a separate repository; when all grading has been completed, we add the marks files to the students repository in a single batch. This ensures that all students receive their grades at the same time, and that the instructor has a copy of the marks files that the students cannot modify. We also frequently add testing data and code to the students' repositories after the assignments have been graded.

Initially, we decided to use CVS to deliver all course materials including lecture notes, example programs, web pages, and JAR files. Although this made it easy for students to obtain all of the course material, the disk space requirements were prohibitive for a large class. We now use CVS primarily for student assignment work. Lecture notes and other course materials are available through a web page, allowing students to be selective about which materials they download.

One possibility that we have not yet explored is to allow markers to add comments directly to students' programs, and commit the modified files. This would allow markers to comment more specifically and directly on flaws in the students' work, and give students higher quality feedback. However, it requires more care and effort on the part of the marker. To benefit from the marker's comments the students would be best served by understanding how to use the "diff" command. The students would also need to learn to revert to a previous version of their assignment to get back the version they submitted.

3.1 CVS Feature Use

We used a relatively small set of CVS features in our second-year courses. Checkout, add, update and commit accounted for nearly all of the operations performed by students. They occasionally used diff and log to examine their repositories. Students rarely reverted to previous versions and never created branches. A few of the more adventurous experimented with tags, and we are now using tags to manage submissions in upper-year courses.

3.2 Issues

CVS was not designed with educational needs in mind. Three specific issues we encountered were

- the need to prevent naive users from damaging their repository,
- the need to prevent students from seeing one another's repositories, and
- the need to prevent students from rewriting their CVS logs to give the impression that they had completed work earlier than they actually did.

3.2.1 Damaging Repositories

In the first two terms, there were several instances of students directly editing files in their repositories, or creating source files in the repository directory, rather than in a checked-out copy of that directory. In these cases, instructors had to do some repair work on the repository in order to get the student back on track, and students lost some or all of the work they had done. These mistakes were due to students not understanding the distinction between master copies stored in repositories, and working copies storied elsewhere. As we became more experienced in explaining CVS to students, this type of problem almost vanished.

3.2.2 Security

Preventing students from viewing one another's work, while allowing instructors and TAs to see it, was more difficult than anticipated due to limitations in Unix's permissions model. We created two types of repositories:

- An individual repository owned by a single student, but with group permissions set to the group that included instructors and TAs (but not the student).
- Team repositories, for assignments done in pairs (or larger groups). The obvious way to implement this would be to create one group for each programming team, and put the students in the team, the TAs, and the instructors in that group. The problem is that on a standard Red Hat Linux installation, a user can only be in 32 groups at a time. (Users can be added to any number of groups, but only the first 32 actually take effect.) This makes it impossible for instructors and TAs to be in the Unix group for all group repositories at one time. The number is configurable, but requires rebuilding the kernel. Our system administrators solved this problem by creating a setuid script that allowed an instructor to edit the group lists, and to change groups. This worked, but was awkward to set up and maintain.

3.2.3 Potential Cheating

The third problem—students editing their version histories in order to fake early submissions—was never fully addressed. We checked out an instance of the repository shortly after the due time to guard against this problem, but TAs still often checked out their own copy to grade, and the snapshot was treated more as a sanity check if needed. We believe that few students are sophisticated enough to find or exploit this security hole, but recognize that once one does, the technique could quickly spread. We are presently investigating strategies such as secure logging of submission times via CVS triggers as a way to close this loophole. A client-server version control system such as Subversion [3] would also solve this problem.

3.3 Use in Subsequent Courses

Since the introduction of CVS in CSC207, we have begun to use it in follow-on courses. In particular, we have used it in "CSC209: Software Tools and Systems Programming". Students learn C programming in this course and all of their assignments are done as individuals, so we did not take advantage of CVS to manage group work. Because we are in the middle of a shift to the new curriculum, only half of the students in CSC209 had previously used CVS in CSC207. Students in CSC209 did not receive as much tutorial information about how to use CVS, but still mastered it quickly, perhaps because their peers were able to help them when they ran into difficulty.

We are also now using CVS in "CSC369: Operating Systems", a traditional operating systems course. We used the OS/161 simulator developed at Harvard [4] and relied heavily on CVS to manage changes to the simulator over several assignments.

Although the OS course at Harvard also uses CVS, students initialize and maintain their own CVS repositories, and use "cvs diff" and "tar" to collect their work and submit it using a traditional submission system. Based on our experience using CVS, we decided to use the same approach to using CVS as in previous courses with a few changes. Students were asked to tag their repository to indicate the version that they expected to be marked. Unfortunately, students frequently forgot to tag their assignments, particularly in the first half of the term, which meant that the TAs needed to do more work to determine what to grade.

The CSC369 students each had their own repository for the first two assignments. They worked in pairs for the remainder of the course, and team repositories were created for each pair. We used branches in the repository to give students access to solution code and to add starter code for subsequent assignments. Given the size of the code base, the changes made to the code over time by the instructor, and the teamwork, it was essential to use a version control system.

By identifying similarities between the process of releasing and maintaining software products and the process of assignment development and submission we were able to use features of CVS such as tags and branches.

4. EVALUATION

CVS can best be evaluated in comparison with the more traditional submission system it replaced.

4.1 Traditional Submission System

The traditional submission system had the following strengths and weaknesses:

Advantages:

- It is possible to prevent students from making submissions after the posted due time. This makes it completely clear to the students that their work was late.
- A web interface allows students to easily submit files from home.
- It is possible to restrict file names, so that the system can enforce some basic submission rules.
- The submission interface is very simple to use.

Disadvantages:

- Students could see a list of files that they submitted, but could not retrieve their submission. This made it difficult for them to be sure that they had submitted the correct files.
- Students could submit their assignments as many times as they wanted, but each subsequent submission overwrote the previous one, so all history was lost.

4.2 CVS

By comparison, CVS had its own strengths and weaknesses:

Advantages

- Students could check out exactly the material they submitted, so that they could verify that their assignment was complete and that everything had been submitted.
- It was possible to deliver starter code in a more immediately useful form than by posting it on the web in plain files or archived format.
- Intermediate commits were available to the instructor, so if something went wrong, or if a student was unable to complete an assignment for non-academic reasons, instructors and TAs had at least partial information to work with.
- When assessing team projects, there was more evidence of which team members did which parts of the work.
- Use of CVS allowed instructors and TAs to commit marks back to the student repositories, which turned out to be an effective way to manage marking of online submissions.
- Managing student assignments through CVS led us to manage TA interaction through CVS as well, giving us more information about the progress TAs were making and a history of their work.
- Using CVS made it significantly easier for instructors to manage a course of four sections spread across three campuses. It was much easier to keep information up to date and instructors were able to consolidate marking in a way that mailing tar files back and forth did not allow.
- Giving instructors control over repositories made it possible to handle difficult situations gracefully, such as team members switching groups because of personal conflicts, or removing permissions from students who dropped the course.
- Finally, there are obvious advantages to having a record of what students did when in cases of suspected plagiarism, or when students and TAs have different opinions about what was completed when. Making students aware of this gives them one more reason to check in early and check in often.

Disadvantages

- Because students no longer need an explicit action to submit their assignments, it was sometimes difficult to determine which version of the assignment was the final version. In particular, we lacked a good mechanism to handle a late policy, which allowed students to submit the assignment late with some penalty. One approach to solving this problem was to require students to notify the instructor if they chose to submit the assignment late. Requiring a separate mechanism is undesirable.
- The first two terms the course was taught, a significant minority of students used CVS inefficiently because they did not understand the underlying model. For example, we found many students repeatedly used checkout rather than checking out once and using update to refresh their local copy. As we gain more experience with the kinds of mistakes students make, we are able to prevent the most common ones.
- Some TAs had trouble determining which version of the assignment they should mark, and often made mistakes (particularly on early assignments). This was due almost entirely to TAs not understanding how to use CVS. We expect these problems to diminish as TAs become more familiar with the tools.
- Although having the instructor set up and manage the CVS repositories for the teams simplifies the work that students need to do, there is more administrative overhead for the instructor when students drop the course or move between teams. We hope that further work on our shell scripts will automate the administrative work involved.

4.3 Anecdotes

A few anecdotes may help give the flavor of what CVS was like in action:

- More than one student was able to recover files that had accidentally been deleted. This reinforced the tool's value, and made it easier for students to complete their work.
- Students appreciated the way CVS helped them synchronize the work they did at home with the work they did on the lab machines. being able to work on their assignments at home. Later in the term, some students even began using CVS to manage assignments in other courses.
- Some students were alarmed by the fact that times in CVS logs are stored in Greenwich Mean Time (UTC). Since Toronto is on Eastern Standard Time, this led them to believe that their assignments would be considered late.
- There were a couple of cases where students asked for help backing out of recent changes they had made. In one case, the student had discovered that he really wanted to change his strategy and wanted to start over again. In another case, a student realized too late that he was not able to complete a new feature and wanted a previous version of his assignment graded. In addition, we had several cases of students asking us to grade a particular version of their assignment because they were not happy with changes they made later. This was a clear benefit to students who were careful about committing intermediate versions of their code, and reinforced the habits we were trying to teach.
- Students frequently came to office hours asking for help debugging their programs. We insisted that they commit the most recent version of their work before seeking help, so that we could check out their work, compile it and run it without needing to log into the student's account. We had complete freedom to modify their code

until we got it to work, and delete our changes without affecting the student's own work. Students then went back to their own workstation to reproduce what they had just seen us do, which ensured that they really understood what had just happened.

- In one case, we realized that only one quarter of the class had checked out the starter code one week into a three week assignment. We were able to present hard evidence to the class that only a small number of students were working on the assignment, and encourage them to get started. This is a trivial example of how the ability to track student progress on an assignment can help instructors catch problems early.

5. USAGE PATTERNS

Since we now had a wealth of data on how students used CVS and could glean some information on how students worked on their assignments, we conducted a study with Keir Mierle and Sam Roweis to use machine learning techniques to analyze the CVS data [8]. We hoped to be able to tell students that we had discovered that students who made effective use of CVS received higher grades on their assignments. In other words, we hoped that we could find evidence that students with good work habits and good time management get higher grades.

Unfortunately, we were unable to discover any features in the CVS data that strongly correlated with high or low grades. We were, however, able to gather some qualitative information from their repositories. For example this is how we discovered that students were using checkout repeatedly rather than using update. To our surprise, there was no statistical correlation between how early students started on assignments, or how evenly they paced their work, and their final grades. We hope that as we (or instructors at other institutions) collect more data, patterns may begin to emerge. We also intend to see whether there are patterns when we look at student usage of CVS in upper-year courses, as the lack of pattern in the second-year course data may reflect nothing more than the fact that students climb the learning curve at different rates.

6. RELATED WORK

The most closely related work on using CVS for assignment submission is part of Penumbra, an Eclipse plug-in intended to provide a simpler interface to Eclipse for novices [9]. Penumbra also simplifies the interface to the CVS plug-in, allowing instructors to use it for assignment delivery and submission. Penumbra hides the complexity of CVS use from the students under an interface that appears as a more traditional submission system, but students can access the full functionality of CVS by switching to the CVS perspective in Eclipse. This makes it possible to use CVS in a first-year course.

The original version of Penumbra used a single repository for the course and each assignment contained a directory for every student. They used Unix permissions to prevent students from accessing each other's work. We believe it is an advantage to our second-year and upper-year students to have their own repositories. It simplifies management of individual students, and gives the student a more realistic view of how they might use CVS in another context. We now have an undergraduate student working to adapt Penumbra to work with CVS repositories that follow our structure.

Using a version control system as a regular part of assignment management leads to interesting possibilities to evaluate and mentor students. The JRefleX project [6] collects and analyzes CVS history data to produce diagrams that can be used to monitor individual and team progress. This "dashboard" allows instructors to identify problems with assignments early enough to help students recover.

7. CONCLUSIONS

A year and a half after we first used CVS for student assignment submission, we believe that it both can and should be introduced in second year. It forces students to adopt good working practice early enough in their careers for those practices to stick; it make it feasible for us to assign team projects much earlier (and to much larger classes) than was previously possible; and it gives the instructors a powerful tool to manage interactions with students, TAs, and each other. The effort required to set it up is relatively small, and with each passing term we have found ways to avoid those problems that do arise.

Other instructors are already adopting CVS for their courses, and we are convinced the end result will be students who are better prepared to develop software to high professional standards.

8. ACKNOWLEDGMENTS

We wish to thank Michelle Craig, Michael Szamosi, and David Daley, our co-instructors for CSC207.

9. REFERENCES

[1] `http://www.cvshome.org`.
[2] `http://www.eclipse.org`.
[3] Version control with subversion. `http://svnbook.red-bean.com/`.
[4] D. A. Holland, A. T. Lim, and M. I. Seltzer. A new instructional operating system. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 111–115. ACM Press, 2002.
[5] A. Hunt and D. Thomas. *The Pragmatic Programmer*. Addison-Wesley, 1999.
[6] Y. Liu, E. Stroulia, K. Wong, and D. German. Using CVS historical information to understand how students software. In *MSR 2004: International Workshop on Mining Software Repositories*, 2004.
[7] S. McConnell. *Code Complete*. Microsoft Press, 2 edition, 2004.
[8] K. B. Mierle, S. T. Roweis, and G. V. Wilson. CVS data extraction and analysis: A case study. Technical Report UTML TR 2004-002, University of Toronto, Augu 2004.
[9] F. Mueller and A. L. Hosking. Penumbra: an Eclipse plugin for introductory programming. In *Proceedings of the 2003 OOPSLA workshop on eclipse tec eXchange*, pages 65–68. ACM Press, 2003.
[10] J. Spolsky. *Joel on Software*. APress, 2004.