Improved computer science assignment submission using the concurrent versioning system

Wayne Piekarski
University of South Australia

This paper presents a novel way of handling electronic student assignment submissions for large computer science courses. While traditional electronic submission systems easily handle students submitting files, these systems do not reflect how software is typically developed in industry. This paper presents how we used the Concurrent Versioning System (CVS) for hundreds of students to develop and submit their software development assignments, and a description of the advantages. Students learn skills that are useful for their future careers, and use a system that makes their assignment development much easier. CVS keeps track of each revision of their program, so students can easily go back to previous versions, and it is impossible to lose their existing work. From the lecturer's point of view, CVS is a common place for all students to submit their work and to record marking information to give back to the student. CVS uses reliable time stamps so it is possible to track how students develop their code, assisting with plagiarism cases. It is also possible to have large teams of tutors all working on marking simultaneously and to keep track of this effectively. We achieved very positive results over the last two semesters with classes of 200+ internal, external, and offshore students. Students expressed that they gained a benefit from using CVS, and other faculty members in the department have expressed interest in using it as well.

Introduction

This paper describes how we have used the Concurrent Versioning System (CVS) to support the electronic submission of assignments for large computer science courses. Existing electronic submission systems are only designed to support the submission of documents, and are not really ideal for handling programming source code. Because the submissions are not rigidly formatted, the marker wastes large amounts of time working out how to run student's assignments. Furthermore, CVS is a tool typically used in industry and we want to give students training with tools that they will be using in the future. CVS has a number of features which make it ideal for teaching, including reliably storing student source code so it cannot be lost, tracking revisions of programs to show an audit trail of their work, and as a way of coordinating the marking process when there are lots of tutors using separate computers to do marking. CVS is designed to work in an environment where students are distributed all over the world, and tutors may work in a variety of locations to do marking. CVS supports the strategic goals of the University of South Australia, which frequently operates courses in parallel both locally and offshore.

In this paper, an overview of current electronic submission technologies is presented. Next, the paper explains how we deployed CVS in the Computer Systems Architecture course at the University of South Australia. We then present results of student feedback, and give conclusions about the success of this project.

Background

This section of the paper discusses the current assignment submission system used at the University of South Australia, and also discusses current tools that are typically used by

software engineers. Assignment submission systems have been extensively surveyed such as in Jones and Behrens (2003), and web-based systems evaluated in Storey et al. (2002). This paper only focuses on the area of computer science teaching.

AssignIT

The University of South Australia currently uses a system known as AssignIT for the submission of all assignments in the university that are in an electronic format (Ryan & King, 2001). Students can submit a single file such as a Word document or a ZIP archive that is then read by a tutor or lecturer for marking. The system uses a web-based front-end which integrates in with the rest of the university's online learning environment, and is quite easy to use for the students. The web-based interface is available anywhere over the internet and so is accessible to all students no matter where they are enrolled.

The AssignIT system works well for simple document submissions that are just viewed or printed, but does not work well for the submission of computer programs written in languages such as Java or C that must be compiled and then executed. In classes with hundreds of students, it is not practical for a marker to click and download each assignment, extract an archive, and then work out how to run the program to evaluate it. This process is time consuming, with much of the time being spent performing tasks that could be easily automated by a computer. One possible solution is to write a program that will perform much of these tasks automatically, but this relies on the ZIP archive files being prepared in a consistent way by the student so that it will be easy to handle. Unfortunately, students rarely follow the given instructions, and therefore it is virtually impossible to write software that can work out how to read all student submissions. Getting tutors to perform a task that should be relatively simple becomes an expensive undertaking with large class sizes, and an automated solution is definitely desirable.

BAGS

At the School of Computer and Information Science, we have previously used a system called BAGS. This is a set of programs that archive up a directory on a Unix system containing source code, and makes it available to the markers for a course. This program is more suitable than AssignIT for source code submission, but can only be used on the one Unix machine and not anywhere on the internet, and is not used for anything else except assignment submission. This system has similar problems with automation as AssignIT, although it is possible to write wrapper programs that can verify the student has submitted their assignment in some kind of proper way. BAGS is quite old and is no longer available, but is an example of an existing source code submission system we have used. Other examples are systems such as that presented in Dawson-Howe .

Concurrent Versioning System (CVS)

The Concurrent Versioning System (CVS) was developed to support the development of software projects (Berliner, 1990; Hung & Kunz, 1992). A versioning control system stores files that are written by developers, and allows them to "check-in' files and attach comments and version numbers to these files. It is possible to browse the history of any file over time, and go back to older versions if newer files are corrupted or deemed to not be as stable as previous versions. CVS is designed to operate over the internet, and allows developers from all over the world to collaborate on the development of software without talking or direct contact.

CVS is commonly used by many popular open source projects on the internet (i.e., the FreeBSD operating system, the Mozilla web browser, SourceForge.net, and the GNU project applications). CVS is also commonly used by many commercial software engineering companies, demonstrating that it is a mature system which is suitable for mission critical applications. There are also many similar commercially-available version control systems, but these are expensive and not freely distributable. CVS is released under the GNU Public License, which means that CVS can be modified and distributed by anyone for free with no

limitations, making it ideal for use by students. CVS has a lot of documentation and tutorials available for free on the internet (CVS Project; Blandy).

When developers use CVS, they gain a number of advantages over their counterparts who only edit files locally. With version control, it is possible to step back in time and look at the source files at any time. Every time a developer changes a file, they must write a comment to a log to describe what the change was done for. These changes can be studied to better understand problems and work with other developers. Computers such as laptops are risky for development because they can be lost or dropped, and computers in general are vulnerable to hard disk crashes, bad floppy disks, viruses, and accidental deletion by humans. Each of these problems can result in a total loss of data, and if the only copy of the software is on that machine all of the work is lost forever. In general, unless a user is a highly technical expert, they will eventually lose some or all of their data at some point in time.

CVS servers are typically located remotely on the internet, and so the copy being edited by the user is not the master copy. The user checks their code into the CVS repository at regular intervals, which makes a copy of their current work along with any comments. Many users can edit the same set of source code at the same time, and CVS will manage these changes and merge them together. Users can work from home, at their place of employment, and in a university student lab without having to bring the source code with them. User machines can be totally corrupted with no effect on the master repository. While it is possible that a CVS server can suffer the same failures as a user machine, this is much less likely because major servers are typically stored in data centres with redundant hardware, backups, and skilled administration staff.

Using CVS for teaching

The author of this paper was asked to take on the Computer Systems Architecture course at the University of South Australia and redesign it. As part of this redevelopment, we wanted to ensure that the course taught the required syllabus but at the same time give experience with common software engineering tools. The author of the paper has been using CVS for a number of years, and based on all the features previously explained in the background section, thought it would be beneficial to both the lecturer and the students. The lecturer benefits by having a rigid framework that assists with marking and tracking student submissions, and the student gains by improved reliability and new skills.

How we teach CVS usage

In the first week of lectures, the students were given a brief overview on how CVS works and the various commands that are needed to operate it. However, the best way to learn a tool is to use it, and so practicals are used to train the students in its use. CVS operates differently than web or paper based submission, and so we were required to train them in its usage. We used practicals to train the students, so that they would be ready for the two major assignments in the course.

The first practical class was a one hour session where students had to check out their personal repository, add some new files, edit them, delete them, and add comments. In the following weeks, the students are required to implement programs that demonstrate concepts shown in the lectures, and they must put these programs into CVS so they can be marked. We allocate 5 per cent of the overall grade to practical submissions, giving the students an effective incentive to do the CVS introduction followed by the practical lessons. It is important that the students learn how to use CVS during the practicals, because if they make mistakes it is better to lose only 0.5 per cent for a practical than for a 15 per cent assignment submission. The students are required to submit their programs in particular directories, and use file names that are named in a very specific way. We found that a few students were confused about if they had submitted their work correctly, and so we provided a small program called checkcsa they could run which would show a list of all the work currently in the CVS repository. Work that is not in the CVS repository is only visible to the student and cannot be marked, so it is important that the student checks their work in properly. It is important to provide tools to the students so that they are able to easily fix their own problems.

Practical marking

The practical marking process is easy to administer because it is fully automated. The marking process simply checks that the specified files are created in the correct directory. Each practical carefully specifies to the student the directory and file names required, and they can use checkcsa to verify they have followed the instructions correctly. Practicals are only awarded a mark if they fulfil all the requirements exactly, and since they are only worth a fraction of a percent each this is not very harsh but teaches a valuable lesson in following instructions. As graduates in industry this is an important skill, because mission critical software used in medical equipment or aircraft controllers requires the software to be built exactly as the specification demands. With our marking scheme, automated tools can mark hundreds of students in less than a minute, with results collated and ready to be pasted into an Excel marking sheet. CVS is designed to interact with external programs very easily - these programs are called shell scripts in the Unix operating system.

Assignment development

The two assignments in the course require the students to develop their program source code using CVS. Unfortunately, many students will try to avoid using it because it is something different and they do not want to change their ways. If they only check in the final version into CVS, all of the problems of traditional development will still exist. To ensure that the students learn of the benefits of using CVS, they need to be coerced into this by making the assessment measure how effectively CVS was used. We award 5 per cent of the marks in the assignments for the use of a sufficient number of CVS checkins, which are evaluated by a tutor during the marking process.

Since students make mistakes following the specification for the assignment, this makes marking more difficult because files may not be named in the correct way or other problems might exist with the program. If each student makes a mistake, this will cost many extra hours in marking time for the tutors, which wastes time and money. We try to ensure that the students follow the instructions by attaching a 5 per cent mark for following all the instructions correctly, and we provide the students a test program that ensures they have done this. The checkcsa program is used to get a copy of the student's assignment out of CVS (not the local version which they may have forgotten to check in) and runs a small set of tests on it. The checkcsa program tests the program in a similar way as the automated marking tool will, and so if it successfully runs then it is highly probable the assignment will work for the tutor as well. The student gets the opportunity to fix up any bugs, the tutor does not have to fix as many problems, and students achieve higher grades overall. The marking system scans the CVS repository later on to allocate the 5 per cent to the student for this part. In the first year of operation, We have observed that the checkcsa testing program has significantly reduced the number of student mistakes – the first run of the course did not have this program. In the first year the marking took the tutors longer and the lecturer dealt with a higher number of complaints from frustrated students.

It is important to realise that the student is always adding their code into CVS. There is no actual submission process because marking is always done on the latest code that is in CVS at the time of the submission deadline. So if students are constantly using CVS for their assignment development, they cannot forget to submit the assignment and they should not have problems at the last minute. If the student forgets to finish the assignment, at least the partially completed work can be marked.

CVS records time stamps for all checkins, and any version at any time can be easily checked out. When the marking process is performed, it may be done many weeks after the submission date. Some students may have been given an extension, and so for some students we need to extract out an even later version. For marking, we checkout a complete copy of CVS from the assignment deadline, plus about two extra hours to cover any last minute submissions. The tutors then mark all of these assignments (this process is described later). The tutors have a list of students who have been granted extensions due to sickness and other circumstances, and these are then marked separately at a later snapshot in time.

Marking process

As mentioned in the previous sub-section, the tutor performing the marking does a checkout of all student submissions, giving the tutor a copy of all the code for marking. Since the assignment specification is quite specific in what is required, automated programs can be used to assist the tutor with the marking process. The tutor runs a testing program for each student to mark, which prepares a report for the tutor to read through. The script automatically compiles the code, checks for any errors, filters out parts of the code such as comments and CVS log entries which the tutor has to read, and so forth. This script saves much valuable time because it is highly mechanical and the tutor does not need to perform this slow process by hand. The generated report is also checked into CVS so the student can read through everything the tutor saw during marking.

The tutor then runs through the program and tests it according to the mark scheme, and generally this works well because most students run the checkcsa program to test it. The marking script automatically prepares a marks template for the tutor to fill in on-screen, and they type in the score for each category of the mark scheme. The tutor also includes full explanations at the bottom so the students have more information about why the marks were deducted. It is important that feedback is given because otherwise the students will question the marks given, generating more work for the lecturer. The marking file is checked into CVS with a log entry, as well as any future remark requests. These logs are useful because it provides an audit trail both for the student and for those running the course. The marks are extracted directly out of the marking template files and into an Excel spreadsheet, removing any ability to make mistakes entering marks, and increasing student confidence in the process.

Plagiarism detection

We use the jPlag software system to detect plagiarism between students – where they copy all or part of the assignment off someone else. The jPlag program does not compare the text of the assignment exactly, but looks for logical similarities between the programs. jPlag is able to detect copying between students even when they have tried to disguise the attempt. Students who have enough knowledge to defeat jPlag probably have the ability to write their own assignment without resorting to plagiarism, making jPlag an effective tool. Since the student submissions are rigidly structured in the CVS repository, it is possible to extract out hundreds or thousands of submissions and put them directly into jPlag in only a few minutes. Since all years of the course are kept in CVS, we can also compare against previous runs of the course as well.

When students have been accused of academic misconduct, the CVS logs can also be used to provide an audit trail of when the code was created. The CVS logs are impossible to edit by a normal student, and each entry is recorded by the CVS server with an accurate time stamp. If one student has a history of changes over a two week period, while another has only one final checkin afterwards, it can be used as evidence to show which student copied off which.

Server configuration

The CVS server runs on a Unix machine in the department and is connected to the internet. A student can use CVS just as easily on-campus as they can on the other side of the world. This is a very important feature, because our university has a number of students who are studying offshore but are marked by tutors in Australia, and also external students who study in a variety of locations and require flexibility in their studies.

Each course in the department is allocated a directory that is named after the course, year, and semester. For example, CSA in 2005 is labelled csa2005s1. We designed it in this way because we anticipated other lecturers would be interested in using CVS as well. The course lecturer is given permission to read and write to all students contained within their class directory, and each student owns their own directory. While students have the ability to make changes to their code, they cannot change the prior history of their work, preserving all auditing abilities for both students and lecturers.

Student feedback

The University of South Australia has a formal feedback process that allows students to rate every course that they are enrolled in, providing valuable comments that are used to evaluate the quality of our teaching. These Course Evaluation Instruments (CEIs) are handed out in the final week of the course during lectures. Other feedback is gained through the use of electronic discussion forums where students discuss the problems they are having.

The Computer Systems Architecture course was rated as being highly successful, and the use of CVS was mentioned in a number of cases as a positive feature of the course. While initially students took a week or two to get used to CVS, most of them saw that it was beneficial to their learning, particularly when they knew it was useful in helping them find a job later on. There were three cases where students completely deleted all of their files on their local machines by accident – CVS was used to check out the original versions and the students were absolutely ecstatic that their work was not lost. This would have been a disaster using any other submission system. Students were also able to work from home using development tools supporting CVS that they were already familiar with, and easily able to synchronise it with the systems on campus. This was not possible before CVS and the students appreciated the ability to streamline their workflow. Many students who initially voiced concern about CVS initially became vocal converts to it after CVS had proven itself to them, particularly in disaster scenarios.

Conclusions

This paper has explained how we have used the Concurrent Versioning System (CVS) for use in teaching computer science students at our university. CVS benefits students because it provides a reliable repository for them to store and develop their assignments, the ability to retrieve previous known versions of files, the ability to work from any location in the world just as easily as a student on campus, and logs that can be used to prove ownership of source code in plagiarism situations. Since they are much less able to lose all their work, there is no need to ask for special consideration when disasters occur.

CVS benefits teaching staff because it can be used to easily track hundreds or even thousands of submissions, and supports automated marking processes that can be used to minimise simple tasks that waste valuable time. With large classes, having more than one person dealing with assignments is required, and CVS helps to support this with its multi-user capabilities and log entries. Other features include being able to easily communicate results back to students, permanently archiving student submissions and tutor feedback, and logs to assist with gathering evidence in plagiarism investigations.

In conclusion, this paper has demonstrated successful results with the use of CVS at our university, with positive feedback from the students. Other academics are investigating the use of CVS in their courses, and we are developing a department-wide system for general student use.

References

Berliner, B. CVS II: Parallelizing Software Development. In Proceedings of the USENIX Winter 1990 Technical Conference, Washington, DC, Jan 1990.

Blandy, J. Introduction to CVS. http://www.cvshome.org/docs/blandy.html

CVS Project. CVS Project Home Page. http://www.cvshome.org

Dawson-Howe, K. M. Automatic Submission and Administration of Programming Assignments. ACM SIGCSE Bulletin, Vol. 27, No. 4, pp 51-53.

Hung, T. and Kunz, P. F. UNIX Code Management and Distribution. In Conference on Computing in High Energy Physics, Annecy, France, Sep 1992.

Jones, D. and Behrens, S. Online Assignment Management: An Evolutionary Tale. In 36th Hawaii Int'l Conference on System Sciences, Hawaii, USA, 2003.

Ryan, Y. and King, R. Impact of the Internet on Higher Education in Australia and Asia. In The E-University Compendium - Cases, Issues and Themes in Higher Education Distance e-Learning, The Higher Education Academy, 2001.

Storey, M. A., Phillips, B., Maczewski, M., and Wang, M. Evaluating the usability of Web-based learning tools. Education Technology and Society, Vol. 5, No. 3, 2002.