AC 2008-927: A SOFTWARE ENGINEERING TOOL FOR MANAGING COURSE PROJECTS

Joseph Clifton, University of Wisconsin-Platteville

Joseph M. Clifton is a Professor in the Department of Computer Science and Software Engineering at the University of Wisconsin – Platteville. He has a Ph.D. from Iowa State University. His interests include software engineering, real-time embedded systems, and software engineering education.

A Software Engineering Tool for Managing Course Projects

Abstract

In the fall of 2006 and spring of 2007, the students in our software engineering project course developed a web-based tool that simplifies tasks associated with project setup, monitoring, and evaluation. We started using the tool in several of our software engineering courses in the fall of 2007.

For faculty, the tool simplifies creation of groups, repositories, phases and activities. The tool can perform periodic checks of repositories using a set of heuristics to determine if acceptable progress is being made. The tool allows retrieval of files in a group's repository without using the SourceSafe or Subversion client. The tool has reporting features to view individual contributions to the project and point deductions due to failure to make steady progress.

For students, the tool allows creation of time estimates and plans, broken down by phase and/or activity. It allows time logging, both manually and via punch-in/punch-out. It allows a student to retrieve files in a group's repository, which is useful when the student does not have access to the version control client. It also has some limited reporting capabilities, such as viewing group summary statistics broken down by individual.

Background

The University of Wisconsin – Platteville has had a BS-SE degree since fall 1999. All of our software engineering courses have a group project component and many require individual projects. For most of the courses, students are required to log time spent on the project, broken down by phase and/or activity. For many courses, students are required to provide time estimates and plans. Furthermore, most of the courses require that students use version control.

The department has tried different mechanisms and tools for time logging. Early on, students were allowed to submit their time logs in text documents. We then experimented with freeware tools and later used spreadsheets with various formats. One problem we noticed for all approaches was that some students would fill in time sheets days after actually spending the time and would guess the actual time spent. To attempt to force the students to provide more accurate time logs, the department developed a spreadsheet version of a time log that had several areas locked. The spreadsheet had a punch-in/punch-out feature that the students were required to use. Manual entries and changes were only allowed through a specific mechanism that marked these entries as such. A student could be required to justify an overuse of manual or modified entries. This worked well for most students but did have some problems for those that wanted to use it on their personal computers but did not have Excel.

For version control, the department uses both SourceSafe and Subversion. Faculty members and students disagree about which is preferable. One disadvantage of SourceSafe is that students cannot use the client from off-campus computers unless they have software that can map to

Novell share drives. One disadvantage of Subversion is that students cannot browse to repositories, but must know the precise name of a repository. For faculty, one issue is the time it takes for creation of repositories, although we have created some tools to semi-automate the task. Another faculty issue is retrieval of work products for review and grading. This can be alleviated by using conventions for repository locations.

Four years ago, several practices were adopted in one of the SE courses with the intent on increasing success rates[1]. These practices included specific work plans, periodically monitored time logs, and periodically monitored version control check-ins. The assumption was that students often fail due to procrastination and that they need to be "encouraged" to start programs early and work steadily toward completion. The "all-night heroic" programming mentality is not consistent with good software engineering. Therefore, the students' version control check-ins are periodically monitored. Students are penalized a small amount each time they fail to show non-trivial progress for more than a specified time interval. As indicated in that paper, a concern was how to address the extra time and effort required for the periodic monitoring and additional grading. It was suggested that tools be used to aid in this monitoring and/or student graders be hired to help with the additional work. Administrators are not eager to support additional funding in the current economic environment.

Class Project

The capstone for the software engineering major is two-semester course sequence. The entire class works on a single project of substantial size. Students are divided into groups and each group is responsible for completion of a portion of the project. Previously, the author had always chosen projects with an industry customer. In fall 2006, the author chose a project with the software engineering faculty members as customers. The initial project goal was to produce a tool to automate the periodic monitoring of student progress on projects. However, because the class had 20 students, it was believed that a more comprehensive, integrated product could be attempted to help address concerns with other tools. In particular, the desire was to incorporate aspects of project planning, estimation, effort tracking, and convenient version control access into the tool.

Development of the tool turned out to be an excellent project for the students. It was challenging and required exploration into new areas for most students. Risks and trade-offs were discussed throughout the project. Prototyping was extensively used in the early phases to flesh out requirements. An object-oriented design resulted in over one hundred classes. There were enough facets to the project that all students were able to find something interesting and remain engaged through the year.

Other papers have chronicled use of specialized software engineering project tools in software engineering project courses, for example see Hawker and Sebern, et al.[2,4] Furthermore, a myriad of such tools have been developed in the open source arena[3,5]. The tool we desired did have project planning, estimation, and effort tracking aspects; however, because the other aspects of the desired tool were rather unique, we decided not to build upon an existing tool such as Software Process Dashboard[3]. It is still unknown whether this was a good decision.

Architectural Constraints

The department has a Linux server and a Windows server. The department also has Novell Access rights to space on the university Novell share drives. University policies as well as other restrictions limit what we are allowed to do. Subversion repositories are stored on the Linux server. Some SourceSafe repositories are on the Novell share and some are on the Windows server. SourceSafe repositories on the Windows server allow web access and can be directly integrated with Visual Studio .NET projects. However, these repositories cannot be accessed via the SourceSafe client from off campus because the area internet provider blocks network drive mapping. SourceSafe repositories on the Novell share cannot be accessed via the web. However, with the Novell client or Novell's NetDrive, drives can be mapped from off campus to allow access. The Novell client and NetDrive are free to students but operate painfully slow when off campus. Furthermore, NetDrive does not work with Windows Vista.

Another issue was the ease with which information could be exchanged between the Linux server, the Windows server, and the Novell share. For example, Novell drives can be mapped from the Windows server; however, rights for Novell drives cannot be set from the Windows server. Novell drives cannot be mapped from the Linux server; however, with a special account, SSH access can be configured.

During the first half of the first semester, the students constructed prototypes for several alternatives. This included web-based clients and stand-alone executables. There were many passionate discussions about what platforms, languages, and tools would be the most appropriate. For example, one student was particularly vocal about Ruby on Rails, but failed to convince other students, many of whom had a brief exposure in another course. Others advocated use of Perl, JavaScript, AJAX, ASP, etc.

The architecture that was finally agreed upon by most students in the class was:

- Separate ASP .NET web-based clients for student and faculty access written in C#
- ASP .NET server residing on the departmental Windows Server written in C#
- SQL Server 2005 database residing on the departmental Windows Server
- SourceSafe repositories residing on the Novell share
- Subversion repositories residing on the departmental Linux Server
- Login validation via the campus LDAP sever
- Version control data "pulled" from the Linux Server via SSH
- Stand-alone executables to retrieve semester start-up data from a university database and populate the tool's SQL database

Tool Functionality for Students

Students access the tool via the secure web link: https://gamma.uwplatt.edu/SE_Tools/. They are prompted for their university e-Directory login name and password. Upon a successful login,

the student is presented with drop-down lists to select the course and project within the course. At that point, the project page comes up, defaulted to the Timelog tab as shown below.

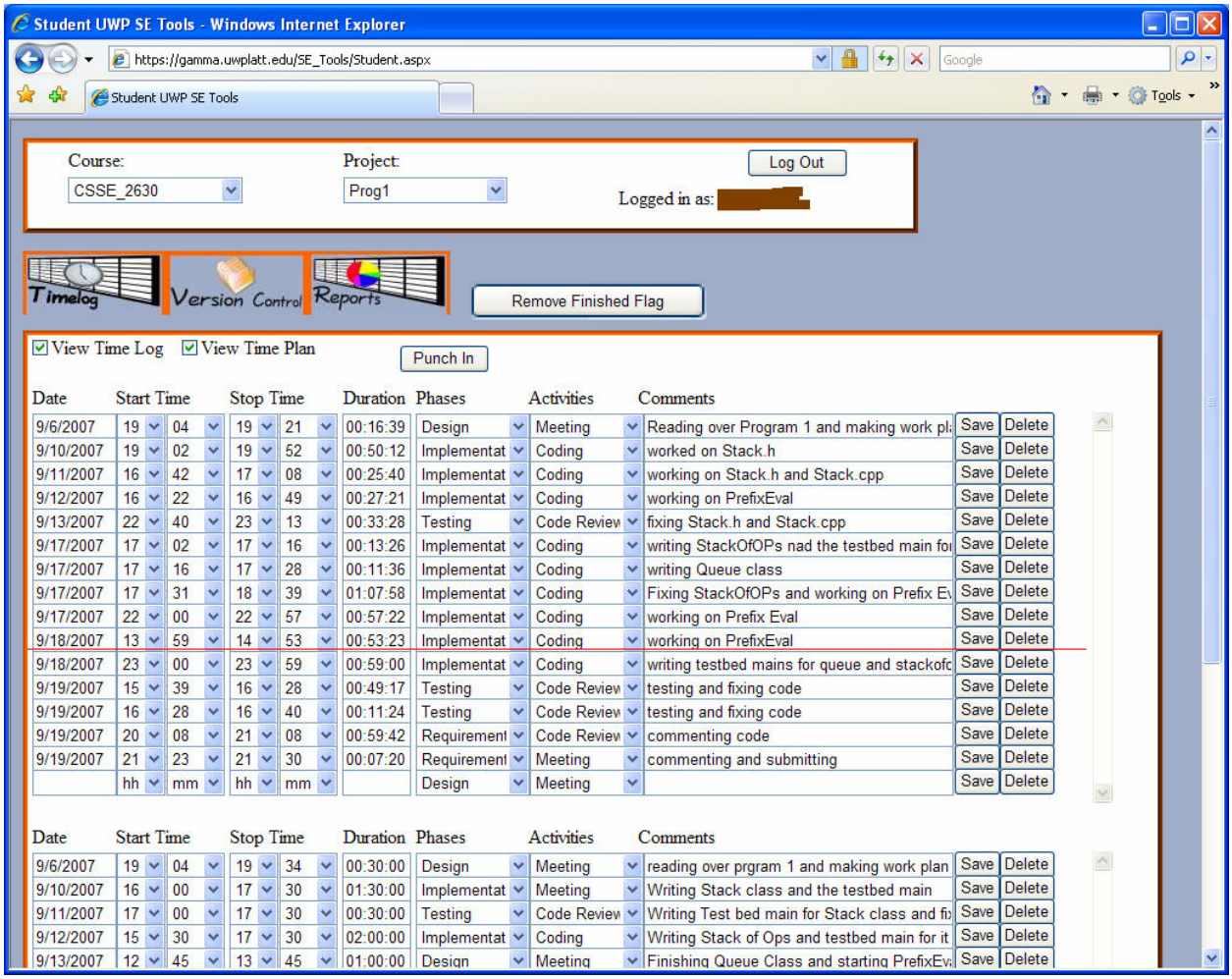

Figure 1: Student Time Log and Plan

Time log entries are generally made using the "Punch In" / "Punch Out" button; however, manual entries can be made and all entries can be changed. Time log entries that are made manually and/or changed are indicated with a red line, such as in the entry shown above for 9/18/2007 at 23:00. Pop-up calendars appear for the Date entry for both the time log and plan. The time entries are via drop-downs but are somewhat cumbersome.

Selection of the Reports tab brings up a page like that shown in Figure 2 below. Time logs and/or plans can be viewed by individual or by group. Summary statistics are broken down by phase and activity for each individual. The activity statistics were cut off in the display below. The report also indicates point "ding" amounts and the dates the "ding" deductions occurred. The students chose the term "ding". As discussed below in the functionality for the instructors, the instructor can set the tool to penalize students that do not make significant progress every few days. Penalizing stops once the project is marked as finished. However, it is assumed that

projects marked as finished will not be modified. If the students later remove the "Finished" flag (see Figure 1), point dinging calculations are applied retroactively.

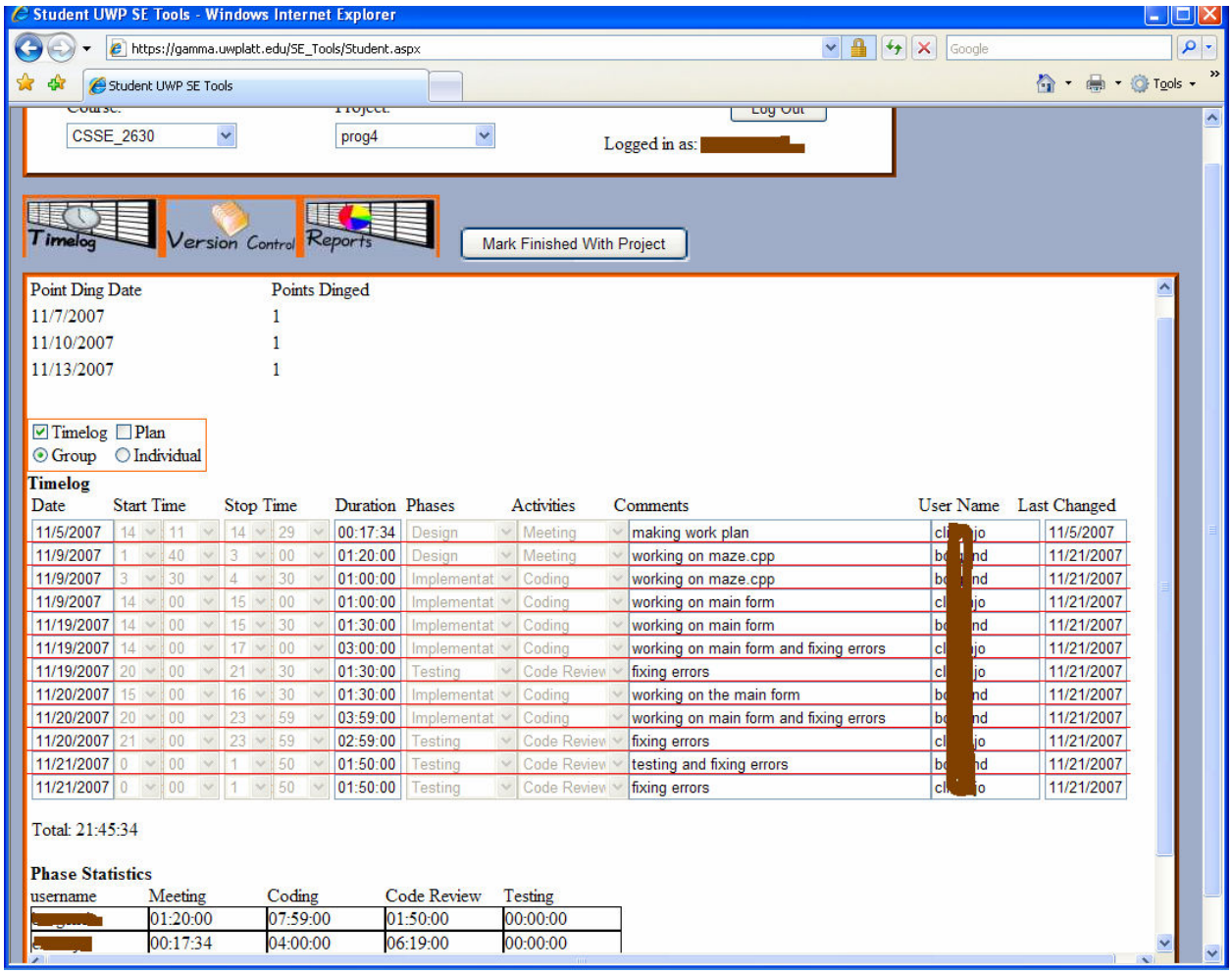

Figure 2: Student Reports

The Version Control tab allows the students to retrieve files from the repositories and is a restricted version of the faculty capability discussed below.

Tool Functionality for Faculty

At the start of the semester, a stand-alone program is executed to obtain course information from a university database. This program captures information for the faculty and students in all software engineering courses and several computer science courses. The information is written to a simple text file. The file has one line for each individual, which includes name, login name, course, section, and a faculty/student indicator. Another stand-alone program is executed that reads this file and populates the tool's SQL database. Thus, when a faculty member logs onto the tool at the start of the semester, all their course and student information is available.

Faculty access the tool via: https://gamma.uwplatt.edu/SE_Tools/Faculty_Admin.aspx.

They are prompted for their university e-Directory login name and password. Upon a successful login, they are presented with a drop-down list to select the course, after which they can select from existing projects or create a new project. Figures 3 and 4 show the page that comes up when creating a new project. The page is admittedly rather busy, so cleaning it up is on the author's to-do list.

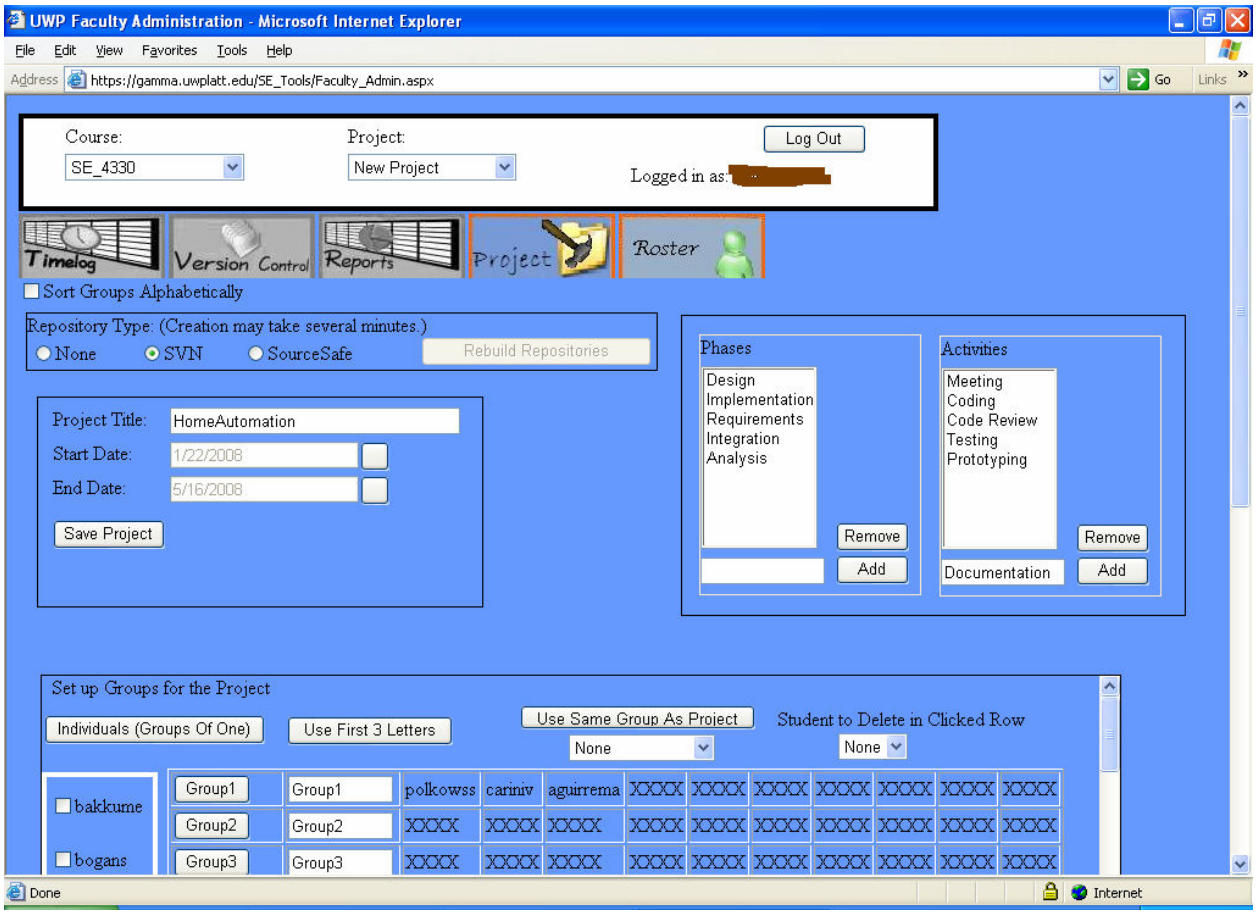

Figure 3: New Project – Top Half

As seen in Figure 3, the instructor chooses the repository type, project title, project start date, project end date, phases and activities. The instructor can set up projects for individuals or groups. To assign students to a group, the instructor selects students from the student list at the left of the panel and then clicks on the particular group button. The name of the group can be the default name, the first three letters from each of the students' login names or any name that the instructor chooses. The instructor can also select to use the same groups as a previous project. As students are added to a group, they are removed from the student list. This allows the instructor to know when all students have been assigned to a group. Thus, creation of a group is very simple. Modifying a group takes extra steps. The instructor must first delete the student from a group by selecting the student number to delete from a drop-down list and clicking on the group button to return the student to the student list. The student can then be assigned to a different group. If all students are removed from a group, the group is removed.

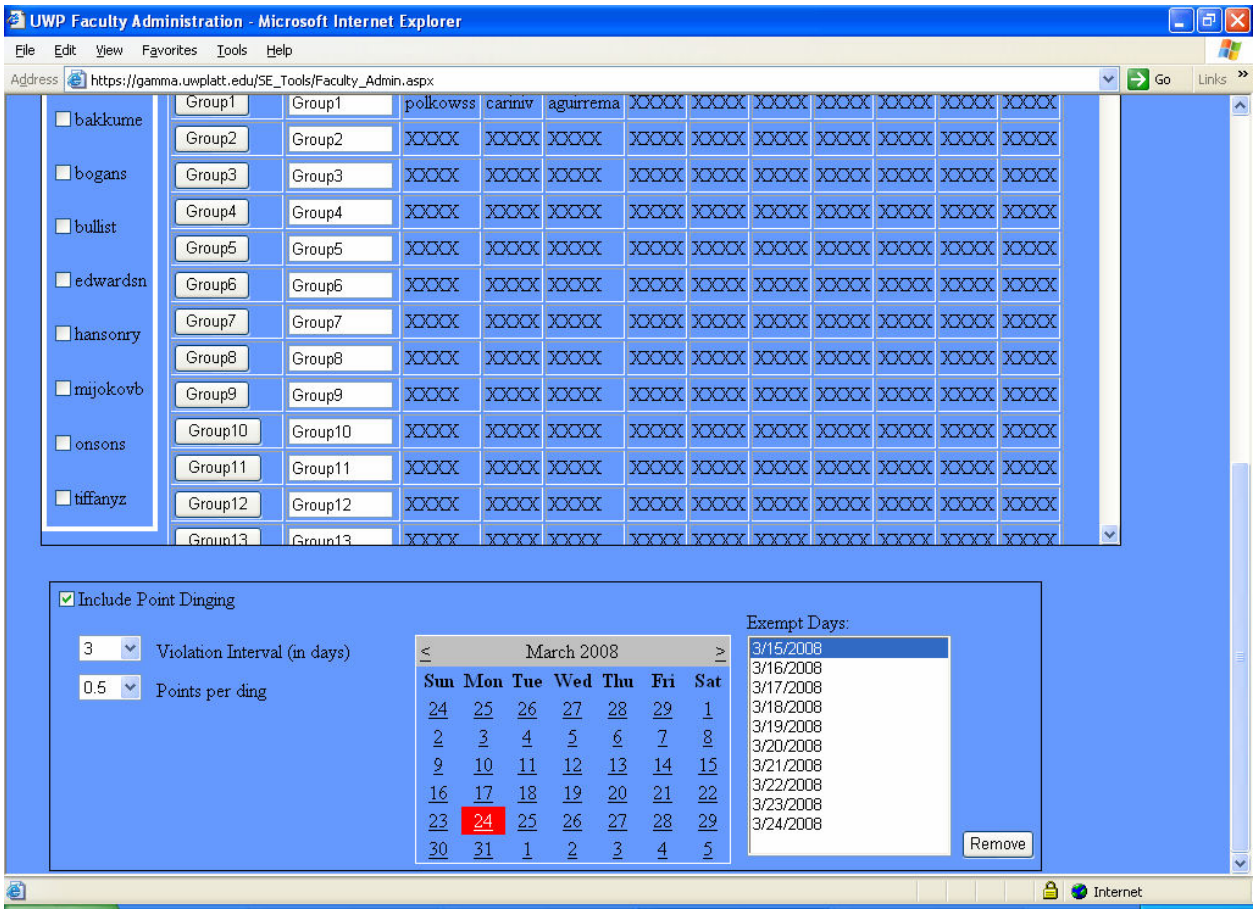

Figure 4: New Project – Bottom Half

Figure 4 shows the mechanism for setting up the parameters of project process checking. The Violation Intervals control is used to specify how often students are required to perform a significant check-in to the repository. The amount per "violation" can be specified as well as days that are exempt from checking. The progress checker is a stand-alone program that runs on the departmental Windows server each morning at about 3:00 a.m. It pulls repository files for each project that has "Point Dinging" enabled. For each file, it determines who checked in the file and when. The retrieved files are parsed and difference comparisons are made based on algorithms presented in a paper by Wu, et al.[6] A set of heuristics are used to compare the latest check-in with the previous check-in to determine if there were significant changes. Thresholds are checked to determine whether an individual should be penalized over a particular time interval for insufficient contribution to the project.

Besides project creation and modification, instructors can add or remove students from the class list via the Roster tab. They can view time logs and plans by group or individual via the Timelog tab, similar to that as shown in Figure 2. Through the Reports tab, the instructor can view reports that include penalty amounts and individual contributions to a group project. Selecting the Version Control tab brings up a page like that shown in Figure 5 below. When the instructor selects the group, the repository for that group is displayed. The instructor then has several options for retrieving files from that repository.

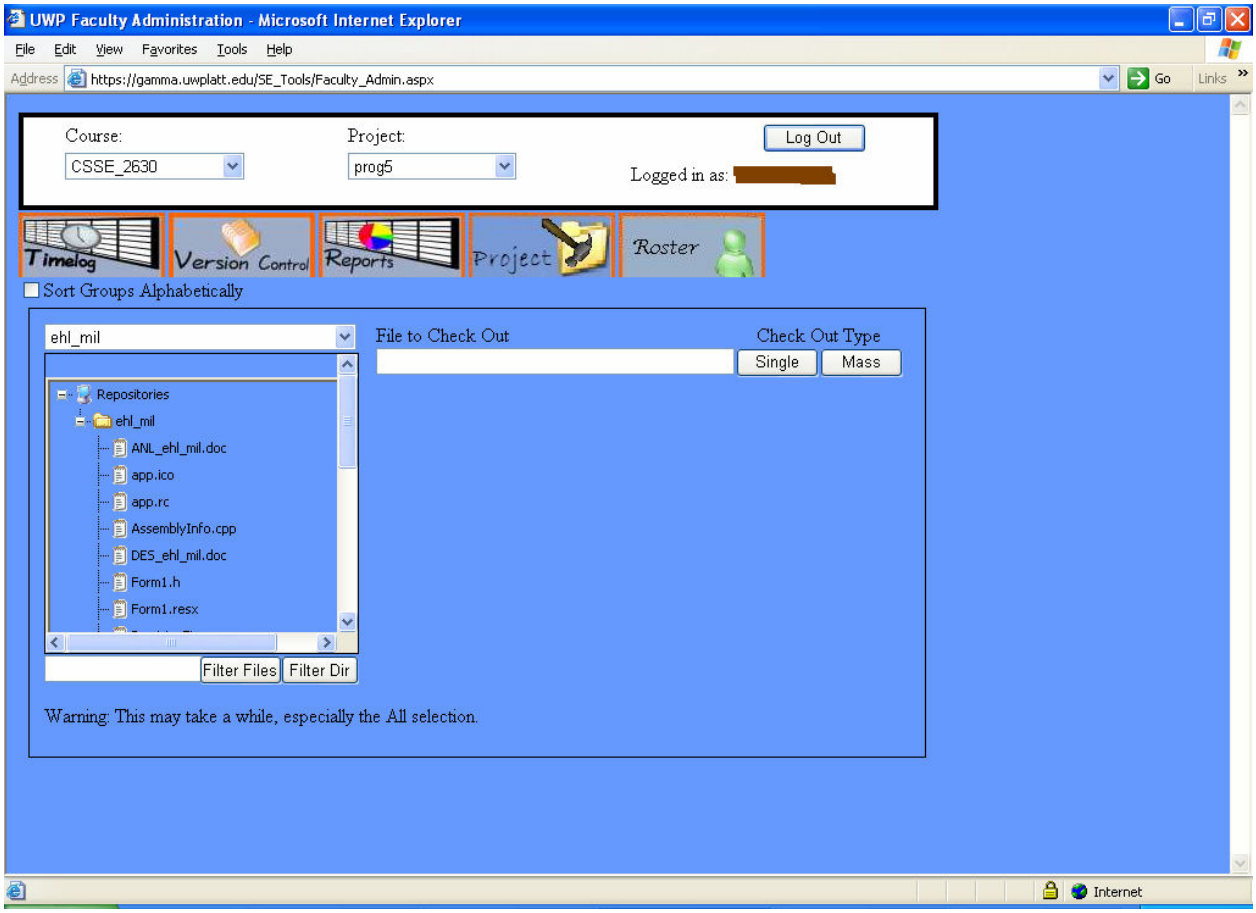

Figure 5: Version Control

Comparisons to Some Existing Tools

One feature of our tool that distinguishes it from others is the progress checker. As suggested by Clifton, failure to start programs/ projects early enough is often a reason for course failure, especially in lower-division courses[1]. By periodically monitoring version control check-ins and applying a small point penalty for lack of progress, more students can be motivated to start early and work steadily toward project completion. Manually checking progress is time intensive. Thus, the progress checker alone is a significant time saver for faculty members that employ this technique. The tool has the added advantage that it automatically sends a warning email the day before a penalty would occur.

Our tool's planning, estimation, and time logging capabilities are similar to those in tools such as Software Process Dashboard. Students can create time estimates and plans, broken down by phase and/or activity. They can log time, either manually or via punch-in/punch-out. Software Process Dashboard also provides defect logging, which is not a feature of our tool. Furthermore, the reporting capabilities in Software Process Dashboard are considerably beyond those of our tool. However, our tool is accessible from a web browser and requires no client software. Software Process Dashboard requires a client installation. Software Process Dashboard is

designed for industrial use and has significantly more features than our tool does. Our tool is designed strictly for academic use.

The version control features of our tool are a limited subset of that provided by SourceSafe and Subversion. The student and faculty member can view contents of a repository as shown in Figure 5. Furthermore, they can retrieve some or all of the files in a repository. The interface is the same for both SourceSafe and Subversion repositories. This functionality is available from a web browser, so the student has access to the repository even when they do not have access to the version control client. However, there are no check-in or checkout capabilities in the current version of the tool.

Another advantage of our tool is that the features discussed above are integrated. This integration is particularly useful given that courses and rosters are populated at the start of the semester. The tool can automatically create repositories when setting up the time planning, estimation, and logging features. Almost all setup can be done with a mouse. The only typing required is for the project name and addition of phases or activities not in the default list.

Compared to tools such as Software Process Dashboard, the weakest part of our tool is the reporting. The reporting was pushed off to the end of the project and students were told to work on reporting features once they finished their other duties. Thus, only a few reports are currently available. The tool allows the faculty member to view individual contributions to the group by size and time. It gives a summary of dates and times students were penalized for lack of progress. It allows visual comparison of the time log versus the time plan.

Next Steps

At the end of the class project in spring 2007, the tool was nearly complete but had several errors that kept it from being functional. The author spent time during the summer of 2007 making the tool functional and the tool was used in several courses during the fall of 2007 and spring of 2008. As one might expect, the author spent many hours during the fall of 2007 fixing errors discovered by faculty and students.

Many modifications and additions could be made to the tool that would make it more useful. Due to speed, the version control capabilities are not widely used. The author has identified some places where efficiencies could be made and will attempt to try those when time is available. The version control could be upgraded to allow full checkout and check-in capabilities versus only file retrieval. Currently, the progress checker only checks text files. It could be modified to check files such as MS Word documents, Rational MDL files, etc. The progress checker retrieves a large amount of information that can be further analyzed to produce more reports. The reporting capabilities in general could be significantly improved. Ideally, the tool could be used to help discriminate between grades given to individuals within a group.

If other universities see value in such a tool, it would need to be ported to a more general platform. The author is considering redoing the project in fall 2008 and treating the current

product as an extensive prototype. The question would again arise of whether or not it should be built upon an existing framework such as Software Process Dashboard.

Acknowledgement

The author would like to acknowledge the students in the software engineering capstone project courses that worked on the tool: Tim Bauman, Bryan Boyer, Aaron Carlson, Nate Edwards, Will Fritz, Kyle Heins, Chris Herrick, Dennis Kalinowski, Gavin Kinsley, Nick Klauer, Scott Messner, Simon Polkowske, Bill Pyne, Dallas Ramsden, Eric Rice, Andy Schaumberg, Kyle Stangel, Chris Treml, Nate Weiss, and Aaron Westerdale.

References

[1] Clifton, J (2006), "Software Engineering Practices Used For Retention?", *Proceedings of the 2006 American Society for Engineering Education*, Chicago, IL, June 2006.

[2] Hawker, J. (2006), "The Collaborative eNotebook: a Collaborative Learning and Knowledge Management Testbed", *Proceedings of the 2006 American Society for Engineering Education*, Chicago, IL, June 2006.

[3] Software Process Dashboard , An open-source initiative to create a PSP(SM) / TSP(SM) support tool, Available at http://processdash.sourceforge.net/

[4] Sebern, M. and Hornick, M. (2007), "Tool Support for Software Process Data Management in Software Engineering Education and Industry Training", *Proceedings of the 2007 American Society for Engineering Education*, Honolulu, HI, June 2007.

[5] Tigris.org, A mid-sized open source community focused on building better tools for collaborative software development, Available at http://www.tigris.org/

[6] S. Wu, E. Myers, U. Manber, and W. Miller, "An O(NP) Sequence Comparison Algorithm", *Information Processing Letters* 35, 6 (1990), 317-323.