ECLIPSE AND CVS FOR GROUP PROJECTS[*]

Ken T. N. Hartness
Sam Houston State University

ABSTRACT

Group projects can be difficult for the instructor to evaluate; students, also, complain about time management and member participation problems. However, working in groups can be valuable for learning course material as well as developing skills vital in the workplace. This paper explores how the open source version control system, CVS, can be used to help students work together with less dependence on synchronous meetings; instructors can also use the system to monitor group member contributions.

INTRODUCTION

Many commercial software projects require contributors who know how to work as a group. Open source projects thrive on this environment. Hopefully, graduating computer science majors will have been exposed to this style of work at least once, often in a software engineering class. However, students could be exposed to this style of problem solving more often, allowing them to learn skills that they can only learn from experience. Many instructors are reluctant to include group projects, especially as a significant part of a class grade, because of the difficulty of evaluating individual achievement. Also, inexperienced students naturally have difficulty planning team activities and handling personality conflicts. As tools arise to assist collaborative software development, hopefully instructors will learn to incorporate more group projects into their classes. CVS is discussed here as a first step toward providing that support.

CVS is a software package that supports team-oriented version control. By constraining students to share their files in a CVS repository, they gain more than a central storage area for their work. CVS maintains a history of changes to files and allows an older version of a file to be restored if later changes are recognized as undesirable; thus, if a student makes inappropriate changes to a file that break a software system, the team members can simply restore the original version before the changes. Also, a log can be created from the version control actions that records the contributions of each member,

[*] Copyright © 2006 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

making it easier for an instructor to confirm claims regarding student participation. In addition, any method that makes the work of the group public (to other members of the group) can also allow the instructor to monitor on-going progress and make suggestions rather than simply wait for the final submission of the work.

CVS supports the creation of parallel branches that can later be merged back together; this technique can be used to allow experimentation with different techniques. This includes the "experimentation" of an eager student who is frustrated with the lack of progress of one or more other members. The response to lack of equal participation can be to create a branch for testing the system; as long as the test modules follow the students' design, later submissions by other members should have the same interface and be easily merged with the finished project.

Eclipse, an open source software development tool, is proposed as a possible development environment for group projects. In addition to its support for CVS and other version control systems, it is a powerful and extensible development environment that can be integrated with other tools, like a chat facility or other remote meeting support software.

BACKGROUND

Group Projects in the Classroom

Sloffer et al [6] suggest that students learn material at a deeper cognitive level as the result of collaborative tasks, especially if some method exists for the professor to interact with the groups at a deeper level than simply evaluating their finished product. One way to encourage this behavior is to require that students keep their current work available in a central repository accessible to the instructor; if at least some of their discussion can be handled by electronic means, then the instructor can choose to participate.

In a study done in Britain [1], groups spread across different universities were able to work together with the help of version control software called BSCW. The researchers discovered that the staff's ability to examine logs of student work was important to the success of the project. In such a distributed collaboration environment, the students' ability to examine each other's work at will was also very important.

Hafner and Ellis also encourage the use of a number of group projects to "foster a greater depth of learning" [3]. However, they also noted some potential problems:

1. good students may perceive group work as a potential threat to their grade;
2. many students lack the social skills necessary for success in group work.

The first problem is alleviated by the existence of a method to verify individual participation. Students can be assured that they will be graded primarily on their individual contributions. Hopefully, the knowledge that their participation will be monitored will encourage lazier students to keep up with their portion of the labor.

As for the second problem, perhaps experience is a good cure. If group projects are easier to evaluate, then more group projects may be assigned, allowing students more opportunities to develop the necessary social skills. The ready availability of each others' work can allow students to critique one another as the project progresses. Also, the instructor, who also has access to electronic discussions and individual contributions, can offer guidance throughout the course of the project and hopefully influence the development of appropriate social skills. Future work may include computerized tools to help guide students and allow them to anonymously evaluate one another.

What is CVS?

Concurrent Versions System (CVS) supports a shared repository of files related to a group project [4]. Derived from RCS (revision control system), it maintains a history of changes to files so that earlier versions may be extracted, if needed. CVS is specifically designed to support groups of individuals who are accessing a central shared repository. It handles mutual exclusion issues and provides some warning to users who are trying to record different changes to the same file; users are usually given the option of cancelling their changes, manually merging their changes with another's, automatically merging their changes, or overwriting the other user's changes. The system supports obtaining files from the repository, making changes, and then merging those changes back into the repository. The more common commands are

1. checkout - make a local copy of a file or project in the repository
2. commit - copy changes from local copy back into the repository
3. diff - list changes made to local copy that make it different from the repository or list changes made between versions.
4. tag - create a new branch starting with some version and allowing independent development to continue from that point while separate changes are made to another branch (or the "trunk").
5. update - merge local copy of a file with a file saved in the repository; can also be used to join two branches. Merging may result in conflicts requiring manual editing of result.

Anything may be recorded in the repository, including design documents and project plans. CVS knows nothing about compilers or programming, in general, and does not limit the contents of the repository.

Normal operation is to create a project within the repository. Each member of the group checks out a copy of the project or at least copies of the individual files on which they will be working. Whenever a stable version is reached, the changes are committed to the repository. Users may use the diff utility to remind themselves of the changes they have made since the last commit.

Eclipse

IBM markets a product called WebSphere Studio Application Developer. Out of a combination of support for open source and a desire to attract a potential customer base for their product, they donated a version of their software to the open source community. The Eclipse consortium now has about 150 member companies [2]. Eclipse represents a general-purpose, extensible software development platform. It is commonly known as a Java integrated development environment, although this is just one of its plug-ins. The IDE supports command completion, incremental compilation, and automatic import generation, among other things. A plug-in for C++ development already exists, and plug-ins for other programming languages are under development.

Eclipse contains limited support for collaborative work by teams. It supports the CVS protocol by default. Users can add a remote CVS repository if the host IP address and path are known. Projects in the repository can be accessed just like local projects. Accessing files in a project checks out a local copy, if necessary. As changes are made to the local copy, the modified file has a > beside it. When the user is ready to commit changes back to the repository, the user simply selects Commit from the CVS menu. The CVS menu will also allow the user to assign version names, create branches, and merge branches together.

The ability to create new plug-ins for Eclipse, as well as its ability to execute external tools within the Eclipse framework, will allow collaborative tools like chat utilities, shared whiteboards, and collaborative CASE tools to be invoked from within the Eclipse IDE.

SUPPORTING GROUP PROJECTS

Students will have to learn a protocol for working together and handling problems that arise. Hopefully, intelligent agents may eventually be created to help monitor interaction and encourage students to follow the protocol. Some sample protocols are included, below.

Designing the project

CVS and message boards support asynchronous work, allowing the members of a group to work independently of one another. However, people find it more difficult to maintain a sense of coordinated work in this environment [5]. Important decisions can often be made more quickly in a synchronous meeting. On the other hand, students with busy schedules often find it difficult to meet at the same time. The following protocol seeks to employ the advantages of both synchronous and asynchronous work.

1. Students agree to at least one synchronous meeting time, if possible, to allow for maintenance of mental focus during initial design phase. (Note that the meeting does not necessarily have to be face-to-face.)
2. Students establish a protocol for calling further meetings, in case progress stalls while using asynchronous communication methods; in any case, several synchronous work times should be attempted around major project points. Students should share an overview of each other's schedules and provide each other with phone numbers and other contact information.
3. At least some synchronous or asynchronous meetings must be conducted in a chat or newsgroup style environment where decisions and participation can be documented.
4. Students will establish a task list for completing the design document and divide the work among the members. The task list will be included in the group's CVS repository.

5. All components of the design document must be submitted to the group's CVS repository, along with the final integrated document.

Coding the project

Hopefully, the hard decisions were all made during the design phase, so students can work independently of one another once they decide how to divide up the work. Students will be able to note one another's progress, which should reduce the amount of redundant effort. The following protocol is intended to support this and other goals described in this paper. Proper use of CVS for group projects will have to be covered, depending on the sophistication of the students, to ensure that the protocol is not a burden to the student.

1. Students will make a task list from the approved design document and assign tasks to each member to complete the project (also to be saved in repository).
2. Where division of labor is made at class boundaries, each member will be responsible for adding new files.
3. Where division of labor is made at the individual method level, a member will be given the responsibility of adding a largely empty file to the repository that will serve as the initial version of the file; each member will check out their own copy of this file and add their method to it. Students will receive instruction on how to merge their work in these cases (in [B], suggests some guidance is necessary on using version control software effectively).
4. Once a project reaches a coherent level, incremental changes must be made on a separate branch until the changes have been tested and/or approved, then merged back on to the trunk.
5. Unless task assignments are formally changed, no member shall modify the main trunk of another member's work; if at any time it is possible that two students are working on exactly the same code, the student not officially responsible for that code must create a new branch.

Failure of a student to participate

A common complaint of group projects is the failure of a student to fully participate in the project. Rather than complaining near the end of the project, groups should have a definite protocol to follow that gives students a chance to correct their perceived lack of contribution and allows the other members of the group to continue working toward completion of the project. CVS has the ability to create independent branches of a project that can be merged with the main branch at a later time; this feature helps support the creation of partial or alternate modules so that progress can continue, then merged with the missing student's efforts if she or he begins contributing once more.

1. Contact the student by two different methods, if possible, and try to resolve any conflicts.
2. If contacting the student fails, consult with the professor before reassigning project tasks as needed to maintain reasonable schedule; contact student with changes.
3. Students may continue a member's work in a branch, but they may not merge the branch with the trunk of the project until such time as it can be verified that the student will not be completing their part of the work.

Statistics Collected with CVS

The CVS repository must be accessible to the instructor (instructor is a member of each group). The instructor can then evaluate submissions directly from the repository. Also, CVS can be configured to log changes and provide an overview of student participation on the project. A simple application can be used to summarize the log and provide statistics on the percentage contribution made by team members.

CONCLUSIONS

Tools should be explored that will facilitate the use of group projects in classes where design techniques are taught. Students will be able to work on larger projects and learn from experience why careful design is important. Using a version control system will help instructors by providing a record of student participation and the evolution of their work. A version control system will help students by removing some of the fear of changing a version of an application only to have it fail to operate. Also, it will make it easier for students to work asynchronously with more up-to-date work than is generally possible without a centralized repository. Hopefully, further development in collaborative tools will allow a rich environment for team projects to be developed, adding to the skills obtained by students before graduating.

REFERENCES

[1] Brereton, O. P., Lees, S., Bedson, R., Boldyreff, C., Drummon, S., Layzell, P., Macaulay, L., Young, R., Student collaboration across universities: A case study in software engineering, *Thirteenth Conference on Software Engineering Education & Training*, p. 76, March 2000.

[2] Daum, B., *Professional Eclipse 3 for Java Developers*, West Sussex, England: John Wiley & Sons Ltd., 2005.

[3] Hafner, W., Ellis, T. J., Project-based, asynchronous collaborative learning, *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, p. 10015a, January 2004.

[4] Harper, D., cvsnt - Concurrent Versions System 2.0.51d, *WinCvs 2.0.2.4*, open source software published at www.wincvs.org (downloaded 9/1/2005), 1999.

[5] Procter, R. A., McKinlay, A., Woodburn, R., Masting, O., Coordination issues in tools for CSCW, *Design Issues in CSCW*, pp. 119-138, London: Springer-Verlag, 1994. [6] Sloffer, S. J., Dueber, B., Duffy, T. M., Using asynchronous conferencing to promote critical thinking: Two implementations in higher education, *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, p. 1083, January 1999.