Group Assignments: Some Lessons Learned and Unlearned

Dr. Annette Schoenberger
Department of Computer Science
St. Cloud State University
avscheonberger@stcloudstate.edu

Abstract

Employers and students alike ask that students graduate and enter their first job with experience working in groups, excellent writing skills as well as knowledge of the technical material in their major. One way to achieve this is to require that students work in groups and that they provide documentation with all their projects. This doesn't work well unless it is accompanied by instruction on how to behave in a group and instruction on what constitutes good documentation. And, this has to be included with the technical instruction.

To this end we have created a course that wraps this new material around a traditional class. The class is now called Software Systems and includes most of the traditional File Structures material with projects.   Instruction also includes program documentation and group behavior.

For the most part it works well. Students who are serious about their work do well and learn more about group dynamics and communication. Work turned in by the groups is exceptional and there is anecdotal evidence that the students do better in group-work after this class. At the same time, problems still exist with students who don't want to do their share or who don't have the technical knowledge to complete their assigned roles.

Introduction

Most students who graduate with a Computer Science major will ultimately work in a group on some large software project. When this happens their ability to work with others can sometimes be as important if not more important than their technical skills. In addition, they are expected to fully document and test their work. This to can be as important if not more important than their technical skills. In fact it is considered a technical skill.

So, just as it is our responsibility to help our students learn the technical skills it is our responsibility to help our students learn how to behave in a group and how to document their work. During the conversion from an academic year based on quarters to one based on semesters we changed the File Structures class to include material on personal work practices, group work and program documentation.

Standard Material

The standard technical material covered in the traditional File Structures class includes indexed files, hashing, b-trees and sort/merge routines all in the context of records on an external storage device. The textbook we use for this material is a standard file structures text [2] that comes with a downloadable project for the students to work on. This is the standard file structures material.

The students do five programming projects that use the code from the text. In the first project the students create classes that will eventually be associated with the records on the files. The first project is fairly simple and is based on prior knowledge. Subsequent projects ask the students to do more complex things with the programs and objects from the previous projects. During the first and seventh week of class material from the Personal Software Process (PSP) is presented [3]. Students are required to log all activities connected with the class and turn in their weekly logs.

The first two projects are individual. At the conclusion the students should have a program that will read and write records to files using a buffering system provided in the book. Students who cannot complete these two projects are advised to drop the class or rethink their work ethic. It is important that they have adequate technical skills if they are going to participate as an equal in the group projects. In previous years when all projects were group projects students without the necessary technical skills or work ethic caused serious problems in their groups.

The final three projects are group projects that involve creating a sequential index, sorting the files, and creating a B+ tree index for the files. Students modify the programs provided with the text to fit the structure of the objects from the first two projects. Each group has five or six members and each member is assigned a role. Half of the grade is based on the quality of the finished project and half is based on how seriously the students take their responsibility to the group. This signals to the student that I expect them to take their group assignments seriously.

Groups

Separate groups are assigned for each of the three group projects.   Each student is assigned one or two roles. Roles are leader, lead coder, coder, tester, documentation specialist. Students assigned the roles of leader and lead coder are not assigned additional roles. All others are assigned two roles. This dual assignment facilitates communication among the group members and seems to hold down the likely hood of students turning into the ‘rogue programmer’.

Group members are required to adhere simple ground rules. Groups are required to meet regularly to coordinate the implementation of the project; all coding and project work other than that specifically assigned as individual must be done in pairs; all project as well as course related activities must be logged in the PSP journals. These rules are designed to keep the students focused on completing the project.

The original work on group behavior rules for this class comes from Theresia Fisher [1]. When I started teaching the class she graciously shared her work with me. I have modified it to fit my personality and experience from teaching other group based classes.

Group Behavior Rules

Regardless of the assigned role students are required to abide by the following group behavior standards.

- Understand that you may need to change your schedule to accommodate the schedules of the others in your group.
- Come to group meetings prepared to participate fully, and do it.
- Work to complete your assigned tasks.
- Coordinate your tasks with the tasks of others in the group.
- Communicate any difficulties you have with the others in the group.

These rules are fairly straightforward. I used to think they were obvious and didn’t need stating but find that students work better in groups when these rules are stated explicitly. Of course not all students adhere to the rules but they do give the others in the group more moral authority when they are trying to work out difficulties.

Roles and Responsibilities

Each of the group roles, leader, lead coder, coder, documentation specialist, and tester has a set of specific responsibilities. The goal is to give specific assignments that will help the students know what is expected of them.

The project leader is responsible for overall success of the project. All communication between the instructor and the group must take place with the project leader present. I do not meet individually with group members other than the project leader. The project leader

- Manages the group toward successful completion of the project
    - Keeps members informed of meeting places and times
    - Chairs group meetings
    - Keeps and distributes agendas, minutes, assignments and deadlines
    - Makes the final work assignments
- Informs the instructor of the progress, obstacles, and challenges facing the group
    - Meets with the instructor once per week
    - Submits written progress reports to the instructor twice per week
    - Talks with the instructor about any problems that come up
- Organizes and runs design meetings
    - Makes sure that all members contribute to the design of the program
    - Makes sure that all members are aware of and contribute to design changes
- Keeps a journal of the project containing
    - Agenda and minutes for all group meetings
    - PSP data for the project
    - Problems or issues confronted by the team along with strategies used to solve the problems
    - Any unresolved issues along with strategies used to solve them

The lead coder and coders work together to implement the groups design. They must work under the supervision of the project leader. So any problems must be brought to the project leader not the instructor. I do not talk to the coders individually. They must come as a group with the project leader so that all hear the same answers. I also insist the coders use paired programming [9]. The use of paired programming helps eliminate the danger of the rogue programmer who presents the rest of the group with an almost complete version of the code the evening before it is due. Or, who makes sweeping changes in the middle of the night. Additional responsibilities include but are not limited to

- During the design phase look for coding difficulties
- Implement the design as agreed to by the group
- Work closely with the testers to make sure the code is adequately tested

The documentation specialist main responsibility is the creation of clear and concise documentation for the project. This includes but is not limited to the design document, the test cases, and the users' manual. I provide copies of these documents from previous project for the students to use as guides. I do meet individually with the documentation specialist to talk about what needs to be in the documentation. During the design phase the documentation specialist must warn the others about those aspects of the design that are difficult to explain. And, at the end, help the testers and the coders prepare their final documents.

Testers design and complete appropriate test cases for the project. During the design phase they must point out aspects of the design that are difficult to test or where it is unclear what the test would be. During the coding phase they work with the coders to ensure the correctness of the code. When the project is complete they work with the documentation specialist to produce the test document.

Using the Personal Software Process

While the students are working on the projects and other course work they are required to us the Watts Humphrey's Personal Software Process [4]. The book comes with lecture notes, quizzes etc. During the first six class periods and five class periods around the middle of the term I give these lectures and quizzes. During the entire term the students are required to log their activities relative to the class.

During the first six lectures the students learn about time logs: what should go in them and how to log these activities. The time logs contain information about time spent on task such as programming, studying, reading the text, doing research etc. At the end of each week the students do an analysis of their weekly activities. My hope is they will see how their work habits are related to their grades. Each week a copy of the analysis and time log is turned in. I look at them and make comments.

During the second set of lectures the students learn to predict how much time an activity will take. Now, in addition to the time logs students plan their work based on past work. At this point some students begin to appreciate the Personal Software Process because they see how it will help them use their time wisely. Others decide it takes to much time. Their work analysis and plans are turned in to the group leaders who summarize the information and submit it with the project documentation.

Students' Use of the Skills

In an effort to find out if the students applied any of this information in subsequent classes we surveyed two of the 400 level CSCI elective classes. Of the 27 students who answered the questions 17 said they were applying the skills. Those who elaborated said they were no longer logging activities but they do attempt to work as a team. They organize meetings, assign tasks and use pair programming. Most said the pair programming was the most helpful. Four of the students said they did not apply the skills. Four said they applied them somewhat. Of these two elaborated to say they used the pair programming techniques. One student said that now that I had reminded him he would in the future.

This is a much better response than I had expected. While the students are taking the class they tend to complain a lot about the group assignments. It is interesting to note that the part they complain the most about is the PSP and that is the part they abandon. Even though they abandon the PSP many said that they now organize their time better.

Conclusion

The class has become more fun to teach since the change. For the most part the students are serious about improving their group skills. They find the information interesting and helpful. They even get a certain amount of enjoyment from the preaching in Watts Humphrey's book.

What I don't know is how much of these lessons they take to other classes. I have noticed that for this class the quality of the work turned in has improved. And, as the survey shows the students are trying to apply the lessons learned to other classes and many find it helpful.

On the other hand, we still have students who can't, won't and refuse to work constructively in groups. It appears that a fixed percentage of individuals in any group never see themselves as part of the problem when the project fails. I do think that by giving each student a roles and behavior guidelines I have helped them deal more constructively with these individuals.

An unscientific pole of the students in a subsequent class shows that students are trying to apply what they learned about working in groups to group assignments. What I need to ask the next time is if they think the others are also trying. That would be the test of success.

References

1. Fisher, T. (Fall 2002) Personal Communication.

2. Folk, M. Zoellick, B. and Riccardi G. (1998). *File Structured; An Object-Oriented Approach with C++*. Reading, MA: Addison Wesley.

3. Gilb, T. (1988). *Principles of Software Engineering Management.* Workingham, England: Addison Wesley.

4. Humphrey, W. (1997). *Introduction to the Personal Software Process*. Reading, MA: Addison Wesley.

5. Keirsey, D. (2002) *Keirsey Temperament and Character Web Site.* http://keirsey.com/

6. Miller, N. and Petersen, C. (1990). *File Structures with Ada.* Redwood City, CA: Benjamin/Cummings.

7. Schach, S. (1999). *Classical and Object-Oriented Software Engineering with UML and C++*. Boston, MA: WCB/McGraw Hill.

8. Weinberg, G. (1971). *The Psychology of Computer Programming.* New York: Van Nostrand Reinhold.

9. Williams, L. and Kessler, R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, V. 43, no. 5, pp. 108-114.

Acknowledgements

I would like to acknowledge the assistance of Theresia Fisher in the preparation of these materials. I would like to thank Don Hamnes and Bryant Julstrom for talking to me about how their students do group work.