AC 2011-2557: TEAMING IN AN ENGINEERING PROGRAMMING COURSE

Cordelia M Brown, Purdue University, West Lafayette

Cordelia M. Brown is an Assistant Professor in Electrical and Computer Engineering, and Engineering Education at Purdue University. She received her Ph.D. in Electrical Engineering at Vanderbilt University, her M.S. in Electrical Engineering at Vanderbilt University, and her B.S. in Electrical Engineering at Tuskegee University. Her research interests include assessment of instructional methods, laboratory design, collaborative learning, and retention and recruitment issues in engineering education.

Dr. Yung-Hsiang Lu, Purdue University

Teaming in an Engineering Programming Course

Abstract

Various formats of teaming have been explored in engineering courses. Engineering courses with teaming have varied from project oriented to capstone design to courses that target first year students. Laboratory oriented courses have also extensively utilized teaming. The formation of teams has also varied from self-selected to instructor selected to computer software team formation tool selected. Outside of pair programming, very little has been studied or reported on the benefits of students working on programming assignments together. In an earlier study, a model for integrating teaming in a programming course was developed. This study seeks to provide the implementation results of a follow-up study of integrating teaming[4,5,6,7,8, 9,10] in a programming course. This study provides results of a survey on teaming administered to students in the programming course. These students participated in a focus group involving questions about teaming in a programming course. The results of this study will be used to improve the model for effectively integrating teaming in an engineering programming course.

Introduction

The course in which the study is conducted is a senior level "Object Oriented Programming using C++ and Java" course[1] at Purdue University.

For this course, students explored topics of object-oriented design and programming, including: (1) objects and classes, (2) inheritance and polymorphism, (3) function overriding in derived classes, (4) operator overloading in C++, (5) exception handling, (6) container classes, (7) multiple inheritance in C++, (8) graphical user interface using Netbeans and Qt, (9) parallel programming, (10) threads, (11) synchronization, and (12) networking. The textbook is "Programming with Objects: A Comparative Presentation of Object-Oriented Programming with C++ and Java" [2] written by Avinash C. Kak and published by Wiley.

The course featured three individual programming assignments, four team programming assignments, three 50-minute midterm exams, four written assignments, in-class quizzes, and a take-home final. The programming assignments were graded through an automated system.

The four team programming assignments centered on the course project. The course project involved programming features of the 3x3x3 Rubik's Cube. The course project was divided into four stages. In the initial stage, teams would program the 3x3x3 Rubik's cube to handle turns and rotations in Java. In the next stage, teams would provide a graphical user interface for a human player in Java. The following stage required teams to switch to C++ in order to solve the 3x3x3 Rubik's cube using computer algorithms. The final stage also used C++ to improve computer algorithms by using fewer steps.

Each student was randomly assigned to a three member team. The make-up of a team changed four times over the semester before each stage of the course project.

In an earlier study[3] involving developing a model for integrating teaming in programming courses, there was evidence that students saw benefits of working in teams when programming. There was also evidence that students valued working with people when programming. In this follow-up study, automatic grading of programs was an added feature to the course. This follow-up study evaluates team assignments, automated grading of programs, and course format for programming courses. It also seeks to improve upon models for integrating teaming in programming courses.

Methodology

A key method for assessing the study was to administer a survey about the team assignments, automatic grading, and course format to the participants. Once the survey had been administered, a focus group was conducted with the participants. The survey was comprised of two components: choice response statements and open response statements. After the administration of the survey, the participants were invited to provide feedback in a facilitated discussion about the topics in the survey. This facilitated discussion provided the facilitator with clarification of the participants' responses. The administrator of the survey and facilitator of the discussion was not the instructor of the course.

Survey Results

There were 24 participants in the survey and the focus group. The results of the survey are as follows:

Choice Response Results

Survey Statement	**Response percentage (%)**
Rank the effectiveness of working in teams in the course.	
Very Effective	16%
Effective	67%
No Effect	13%
Somewhat Ineffective	4%
Very Ineffective	0%
No Response	0%
Rank the effectiveness of automatic grading in the course.	
Very Effective	21%
Effective	67%
No Effect	0%
Somewhat Ineffective	4%
Very Ineffective	8%
No Response	0%

Rank the effectiveness of the course format for the course.	
Very Effective	25%
Effective	63%
No Effect	4%
Somewhat Ineffective	4%
Very Ineffective	0%
No Response	4%

Open-ended Response Results

Team Assignments

What did you like best about working in teams?
The major themes that evolved were: (1) division of workload, (2) simulates industrial programming environments, (3) exchange ideas, (4) meet new people, (5) brainstorming, (6) eases pressure, and (7) encourages efficiency.

What would you like to change about working in teams?
The major themes were: (1) self-selection of partners and then shuffling of pairs, (2) less frequent changing of team members, (3) nothing, and (4) increase size of project.

How is your overall experience working in teams? Explain your response.
Most of the responses to this question were generally positive. There are a few responses indicating a poor experience.

Explain your thoughts about the idea of changing team members multiple times throughout the semester while working of the same project.
The major themes were: (1) meeting other students, (2) difficult to deal with different code bases, (3) three member teams were too small, (4) must come to a consensus on what to continue working on as a team, (5) must understand project to continue working, (6) see different approaches, (7) not "stuck" with people, (8) exposure to various views, (9) for competition, changing teams is a bad idea, (10) changed teams too frequently, (11) no improvements needed, (12) more accountability, and (13) less phases.

Do you think changing members of the team helped you write better code because your code had to be understood by different team members in different stages? Explain why or why not.
The major themes were: (1) initially, yes, (2) not if the code worked, (3) quality of code was not a priority, (4) it helps you organize the structure of the coding, and (5) must add more comments.

Do you think changing members of the team helped you read code better because you had to understand the written code from different team members in different stages? Explain why or why not.

The major themes were: (1) it helps you read code quicker, (2) learn new techniques, (3) learned to translate from Java into C++, and (4) reading more code increased my ability to read code.

What are your suggestions about team assignments in future offerings of the course?
Some of the themes from the suggestions are: (1) focus on several project themes versus one, (2) keep everything the same, (3) two phases, (4) keep random assignments, (5) more individual accountability, (6) more games, (7) complex team assignments, (8) competitive within teams, (9) competitions between teams, (10) rotate through project leaders who are responsible for assembling teams, (11) rotate team members less, (12) similar parts of projects should be done by the same team members, (13) have more time allotted for each stage, (14) difficult to code Rubik's cube in phases, (15) smaller teams or larger projects, (16) use video games for course project, (17) keep same language throughout, (18) allow for self-selected pairs that are randomly placed in different teams, and (19) no less than four team members.

Automatic Grading

Did you find automatic grading helpful?
From the responses, 70% reported yes automatic grading was helpful, 13% reported somewhat helpful, and 17% reported not helpful.

What did you like best about automatic grading in the course?
The major themes were: (1) viewing where solution failed and being able to correct, (2) being able to work towards a perfect solution, (3) instantaneous feedback, (4) know the score before deadline, (5) opportunity to improve solution before its due, (6) hints about where problems are in the solution, and (7) unlimited submissions before deadline.

What would you like to change about automatic grading in the course?
The major themes were: (1) provide more specific hints about each test case, (2) nothing, (3) improve details of the feedback, and (4) lack of opportunity to explain work.

How many times did you submit the same assignment?
The range of the number of submissions of the same assignment was from three to twenty. The average was between seven and eight. It was frequently noted that the first individual programming assignment was submitted between fifteen to twenty times. It was also frequently noted that all other assignments were submitted five or less times.

Did your scores improve steadily? If not, what did you do? Did you go to the instructor's or TA's office hours?
Most reported a steady increase in their scores. For the participants that reported that their score did not improve, they reported seeking help from the instructor or TA.

Do you think the instructor should set limits on number of submissions (say, 10 times) so that students would not try aimlessly?

The major themes were: (1) there should not be a limit because it does not simulate a real world setting, and (2) there should be time limits between submissions.

What problems did you encounter with the automatic grading tool?
The major themes were: (1) highly specific feedback, and (2) not enough detail provided in the feedback.

Were the grade reports informative for you to improve your program?
From the responses, 63% reported yes the grade reports were informative in improving their program, 33% reported the grade reports were somewhat helpful in improving their program, and 4% reported the grade reports were not helpful in improving their program.

Course Format

What are your overall thoughts about the course format?
The major themes were: (1) format was good, (2) use more Powerpoint slides, (3) course was layed out well, and (4) well structured.

Conclusions

The conclusions for this study are drawn from the survey, focus group, and analysis of the teaming model in programming courses. The survey provided insightful details about what works best with the teaming assignments, automated grading, and course format. The survey also reveals information on things that could be improved in each of these areas. For teaming assignments, the survey revealed that students thought that the best aspects of the teaming assignments were opportunities to divide the labor, brainstorm, exchange ideas, meet new people, simulates an industrial programming environment, and that it encourages efficiency. The survey and the facilitated discussion revealed that there were many thoughts on how teams could and should be formed. A key thread through many of the suggestions on forming teams involved having some form of consistency among team members. Overall, the automated programming grader was viewed to a positive addition. This additional resource allowed students to receive immediate feedback about the submitted program. The survey and focus group provided evidence that the course format is effective for programming courses. The results from this study will be used to improve the teaming model for programming courses and improve future offerings of the course.

Bibliography Information

[1] Course website: https://engineering.purdue.edu/OOSD/F2010/index.html.

[2] Kak, A. C., "Programming with Objects: A Comparative Presentation of Object Oriented Programming with C++ and Java", John-Wiley, March 2003.

[3] Brown, C. and Y-H. Lu, "Integration of Real World Teaming into a Programming Course". In Proceedings of 2010 American Society for Engineering Education Annual Conference & Exposition, July 20-23, 2010.

[4] Smith, K. A., "Teamwork and Project Management", 3rd Ed., McGraw-Hill, 2007.

[5] Williams, L. and R. Kessler, "Pair Programming Illuminated", Addison-Wesley Longman, 2002.

[6] Adams, S. G., "Building Successful Student Teams in the Engineering Classroom". Journal of STEM Education. July-December. Auburn, AL., 27-32, 2003.

[7] Morris, D., "Automatic Grading of Student's Programming Assignments: An Interactive Process and Suite of Programs". In Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference, S3F-1 – S3F-5, 2003.

[8] Ala-Mutka, K. M., "A Survey of Automated Assessment Approaches for Programming Assignments". Computer Science Education, Vol. 15, No. 2, 83-102, 2005.

[9] Lui, K. and K. Chan, "Pair Programming Productivity: Novice-novice vs. expert-expert, International Journal of Human-Computer Studies, 64(2006), 915-925.

[10] Hannay, J., T. Dyba, E. Arisholm, and D. Sjøberg, "The effectiveness of pair programming: A Meta-analysis", Vol. 51, Iss. 7, July 2009, pages 1110-1122.