AC 2010-1820: INTEGRATION OF REAL WORLD TEAMING INTO A PROGRAMMING COURSE

Cordelia Brown, Purdue University

Yung-Hsiang Lu, Purdue University

Integration of Real World Teaming Into A Programming Course

Abstract

Historically, teaming experiences for engineering students has primarily been found in first year engineering courses, design courses, and laboratory courses. Occasionally, other types of engineering courses integrate teaming as a part of some of the course projects. In this paper, we are reporting our findings of integrating teaming into a programming course. This study examines team projects and team interaction in a senior-level programming class. The course project spanned the entire semester and is divided into four stages. The students have different project team members in different stages and each team includes 3 to 4 students. The students have to use their own code based from the previous stages. This course involved object-oriented programming covering both C++ and Java with an emphasis on their similarities and differences. Students could choose the languages for their projects, and may change languages for different stages. The project implemented extensions of the popular computer game Tetris®. The extensions included: (1) allowing pieces of 5, 6, or 7 squares per piece, (2) developing algorithms to automatically rotate and place pieces, and (3) competing against other teams' algorithms through a network.

This study gave students opportunities to handle team changes that are common in the workplace due to various business conditions. The changes require robust designs in their programs and clear documentation. Moreover, the instructor encouraged students to use Java for the first stage by giving two small-scale programming exercises in Java. In the second stage, students were encouraged to use C++ (with Qt library for GUI) because Qt has an open-source Tetris® program.

This study seeks to develop a model for integrating teaming[3,4,5,6,7] in programming courses. To assess the effectiveness of the model, we administered a survey and conducted a small group analysis with the students in the course.

Introduction

This study involved students in an elective senior-level “Object-Oriented Programming using C++ and Java” course at Purdue University[1]. In this course, the students learned the concepts of object-oriented design and programming, including: (1) class and objects, (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) client-server networking, and (10) multithreading. The textbook is “Programming with Objects: A Comparative Presentation of Object-Oriented Programming with C++ and Java” by Avinash C. Kak published by Wiley[2]. All lectures were recorded in advance using Camtasia Studio. This tool performed screen capture with narration so that the instructor could show slides, websites, code, and demonstrations of program execution.

The course included three individual programming assignments (student had to choose two different programming assignments to complete), ten laboratory exercises, a four stage team assignment, three exams, and a final exam. There were three lecture sessions (50 minutes per session) scheduled per week. The lectures were recorded on video. Students were encouraged to view the lecture according to a schedule for the course materials. There were two lecture videos per week. The instructor was available in the classroom during lecture hours and the teaching assistant was available in the software laboratory during the lecture hours, three times a week. Students could use the lecture hours for programming assignments, lab exercises, discussion, or asking questions. Additional office hours could be arranged by email.

The lectures, assignments, and lab exercises were recorded in separate videos. Questions were embedded in the lectures as self tests. The lab exercises were closely integrated with the programming assignments. There were practice questions at the end of each video clip. The source code used in each lecture was available separately so students can study the source code.

Programming Exercises/Assignments

The programming exercises/assignments were all based on computer games. The programming exercises/assignments for the students were subdivided into three categories: lab programming exercises, individual programming assignments, and the team programming assignment.

Lab Programming Exercises
The students were given monthly due dates to complete the ten lab programming exercises. By the end of the first month, students were to have at least three of the lab programming exercises completed. By the end of the second month, students were to have at least seven of the ten lab programming exercises completed. By the end of the third month, all of the lab programming exercises had to be completed.

Individual Programming Assignments
There were three individual programming assignments for students to choose two different assignments to complete. The three choices were: Breakout, Pacman, and Space Invaders. Students were given additional credit if they created a one-minute or longer demonstration of the game and post it for the public to view.

Team Programming Assignment
The team programming assignment implements extensions of the popular computer game Tetris®. Team members were randomly assigned to each team by the professor. The team assignment was divided into four stages, with increasing degrees of flexibility and freedom for creativity in design and implementation. The goals of the team assignment were to: (1) have teams design and implement a non-trivial program in multiple stages, (2) provide students with experience in working in different teams, and (3) provide students with experience in handling team changes. The four stages were: Generate Tetris® pieces (Stage 1), Interactive Super ® (Stage 2), Two-player Super Tetris® through

the Internet (Stage 3), and Competition (Stage 4). Students could use Java or/and C++ for the team assignment. Computer languages could be changed at different stages, but in each stage, every team member must use the same language. In the original Tetris®, there were seven pieces with four squares each. In Super Tetris®, each piece must have 4, 5, 6, or 7 squares. A two-player Tetris® is an extension by allowing two people to play and compete. To demonstrate this stage, two people should be able to play Tetris® in side by side play panels. When a complete horizontal line is completely filled, this line is eliminated and the line, except the last filled square (or squares), is sent to the bottom of the opponent's play panel. To infuse a real world teaming experience, after each stage, students worked with different team members. Students were required to use their own code from the previous stages.

Methodology

As a means to begin to analyze the effectiveness of integrating real world teaming in a programming course, a small group analysis of the course was conducted with students from the course. During the small group analysis, students were given a survey to complete. The survey contained choice response statements and open response questions. This survey contained items about teams, various aspects of changing team members frequently, Super Tetris® team project, team projects, and the course format. Once the students completed the survey, they were invited to discuss the items with one to two of their nearest peers. After a short peer discussion, most of the time was devoted to a facilitated discussion around items in the survey to help clarify responses to survey items and for students to elaborate on their responses. The facilitator of the discussion was not the instructor of the course.

Survey Results

There were 46 students enrolled in the course. Of the 46 students, 11 volunteered to participate in the small group analysis.

Choice Response Statements

Statement	**% of respondents**
Rate the effectiveness of working in teams for the course	
Very Effective	*9%*
Effective	*82%*
No effect	*0%*
Somewhat Ineffective	*9%*
Very Ineffective	*0%*
Rate the effectiveness of the team assignment for the course	
Very Effective	*27%*
Effective	*73%*
No effect	*0%*

Somewhat Ineffective *Very Ineffective*	*0%* *0%*
Rate the effectiveness of the course format *Very Effective* *Effective* *No effect* *Somewhat Ineffective* *Very Ineffective*	 *27%* *46%* *9%* *9%* *9%*

Figure 1. Survey results of choice response statements

Open Response Statements

Teams

What did you like best about working on teams?
There were five major themes that emerged from this question: (1) generate multiple ideas, viewpoints, and skill sets; (2) similar to working in a company; (3) division of labor; (4) project did not seem insurmountable; and (5) learning different people.

What would you like to change about working on teams?
There were eight major themes surrounding changes that could have been made involving working in teams: (1) appropriate incentives could be provide for code quality and team contributions; (2) access to previous team assignments before starting new team assignment; (3) self-selected teams; (4) changing teams as an option and not a requirement; (5) less frequency in changing teams; (6) base team make-up on amount of time and effort devoted to class; (7) extend the size of the team to 4 or 5 members; and (8) more accountability for individuals.

How is your overall experience working on teams? Explain your response.
There were eight major themes about their overall experience of working on teams: (1) there is variation team members' contribution; (2) gained an appreciation of random team assignments; (3) learned more about different classmates; (4) good experience; (5) keeps motivation up; (6) some teams were too disorganized; (7) more time spent on coordinating teams and synchronizing code than actual development of code; and (8) no problems working in teams.

Explain your thoughts about the idea of changing team members multiple times throughout the semester while working on the same project.
There were fourteen major themes surrounding the students' thoughts of changing team members: (1) chance to change teams if things are not going well; (2) for competition purposes, multiple teams would have similar issues; (3) similar to real office environment; (4) learn more people; (5) interesting and exciting; (6) frustrating; (7) forced teams' members to decipher new code at each change; (8) did not build as much

camaraderie as you would having the same team members throughout the semester; (9) not every team started with the same advantage of using really good code; (10) larger teams are needed when changing; (11) interesting experiment, seems to work; (12) changing team members is a great idea, but using the same project is not; (13) helps you learn to adapt to change; and (14) teaches you to interact with other personalities.

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.
There were nine major themes around changing team members and writing code: (1) must write better code because your name is attached to it; (2) not enough time to write better code; (3) allows you to see other ways code could have been written; (4) no, ended up working on something new or something familiar; (5) no, people tend to maintain their coding style; (6) no, little incentive to write better code; (7) no, never thought about it; (8) no, it forces people to document more, and (9) yes, encourages you to write better code because someone else would be using it.

Do you think changing members of the team helped you read code better because you had to understand code written from different team members in different stages? Explain why or why not.
There were seven major themes around changing team members and reading code: (1) yes, over the course of changing teams, you are reading at least 10 people's code; (2) sometimes, learned different methods from reading others' code; (3) no, if you can write code well, then you can read it, taking time to decipher others' obscure code multiple times, wastes time; (4) no, code is code, there is only one way to read it; (5) perhaps, its more frustrating to try to understand what the code was doing so it could be modified; (6) no, it's a struggle to read other people's code, even if it is documented; and (7) didn't get a chance to read someone else's code.

What are your suggestions about team assignments in future offerings of the course?
Some of the suggestions about team assignments for future offerings of the course are: (1) after the first team assignment is complete, share all teams' codes with the class, so we can learn and use the better one for the upcoming team stage; (2) pick your own team, (3) keep the same teams all semester; (4) liked how the Super Tetris® was broken up into phases; (5) more time dedicated to later stages and less time to earlier stages; (6) change project completely each cycle or give all teams access to same code; (7) identify "slacker" teammates and do not place them on teams with hard workers; (8) create larger teams of at least 4 or 5 people; (9) give each team has a different team project; (10) best documentation criteria should be awarded in all steps; and (11) include individual accountability.

<u>Team Assignment</u>

Do you think Super Tetris® is appropriate for this course as a team project? Please explain your answer.

From the student responses to the appropriateness of Super Tetris® as a team project, there emerged five themes:
(1) Yes, great way to involve basic coding skills with artificial intelligence and networking; (2)Yes, with the exception of artificial intelligence; (3) Yes, making a game provided interest; (4) Yes, first stages were appropriate, but artificial intelligence seemed out of the scope and team competition seemed unfair, (5) Yes, because it has all the features of great games (artificial intelligence, multiplayer/network play, and scoring system), and (6) Yes, it was complex enough and was able to be divided in several stages.

What are the characteristics of appropriate projects for team projects in this course?
The suggested characteristics of appropriate team projects are: (1) competition for bonus only; (2) including games; (3) artificial intelligence; (4) usage of logic; (5) complex enough so that team effort is needed; (6) some aspect of fun; (7) easily written in C++ or Java; (8) projects with phases; (9) projects using a network; (10) orthogonal modularity; (11) modules should be independent within a stage, (12) score independent of other teams, and (13) moderately complex.

Please recommend several projects that have these characteristics.
Some of the suggested projects with the about characteristics: (1) tower defense game; (2) other games that do not get too focused on particular algorithms (for instance Breakout can be too focused on collision detection, depending on implementation); (3) chat client; (4) file sharing program; (5) any game over a network; (6) non-game project; (7) any sports related game (football/soccer); (8) checkers, (9) 3-D Tetris®; and mini-arcade that has several single games.

Course Format

The instructor gives lectures in the classroom at the same time when the TA is holding a laboratory session in the computer laboratory. You can attend either learning session. Do you think this arrangement is helpful to you?
The comments provided to this item were mixed. There were strong reactions to the benefits of the able to attend the learning session that best met your current needs. There were also strong reactions to not being able to attend both in a given day. There was also a positive reaction to have both the professor and the TA swap places in order to get different perspectives.

What are your overall suggestions for this course?
The overall theme of having teams for the course is consistent. The means for team member selection, changes in team composition, and frequency of changes in team composition varies widely. Suggestions for providing using the team evaluations to give different members, different grades within the team based on their contributions were provided. There were mixed opinions about the use of grading in the competitions. There was also a suggestion of requiring updates from teams between stages. Among those surveyed, another overarching theme was enforcing more accountability for individuals.

Observations and Conclusions

The results from the small group analysis, conclusions, and future work on the model for integrating real world teaming are as follows. The small group analysis helped provide some insight on the integrating of real world teams into a programming course. From the students' response, there is evidence that students saw benefits of working on teams. It is evident in the responses involving teams provide an opportunity to generate multiple ideas, listen to different viewpoints, and utilize different skill sets. There is also evidence that there is value in working with people. Through the survey, it can also be observed that there are distinct differences on how the teams should be formed. In future iterations of the model, efforts to employ team formulation tools can be incorporated. Through the discussion, many of the students felt it important after each stage was completed that full set of each teams code be shared with the entire class. This would provide students with more opportunities to read code. In future offerings of the course, team evaluations with individual accountability will be used.

Bibliography Information

[1] Course website: https://engineering.purdue.edu/OOSD/F2009/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] Friedman, R. and D. Fadi, "Innovation and Education in the Digital Age: Reconciling the Roles of Pedagogy, Technology, and the Business of Learning, IEEE Transactions on Engineering Management, November 2003.

[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] Oakley, B. A., D. H. Hanna, Z. Kuzmyn, and R. M. Felder, "Best Practices Involving Teamwork in the Classroom: Results From a Survey of 6435 Engineering Student Respondents", IEEE Transaction on Education, Vol. 50, No. 3, 266-272, August 2007.