NISOD 40 years of celebrating excellence in teaching, learning, and leadership!

INNOVATION ABSTRACTS

June 21, 2018 ♦ Vol. XL, No. 21

Legacy Edition

Divide-and-Conquer Programming Revisited

When I wrote my 1998 *Innovations Abstracts*, "Divide-and-Conquer Programming: Using Interdependent Teams," I was an adjunct professor, only teaching face-to-face classes. In the article, I argued that student group projects that used the divide-and-conquer approach—where students in each group divided project work among themselves and integrated the resulting pieces into a finished product—were an effective and realistic approach to teach software development. Today, I teach full time and I still believe that student group projects enhance students' learning experiences. However, as I have shifted to mostly teaching online courses with no face-to-face component, I've had to adapt the divide-and-conquer student group project approach to the realities of virtual teaching and student interaction. Instead of grouping students into teams of three or four for class projects, I now design projects that only require communications between two students at a time.

Initial Efforts

My initial efforts to apply what I did in the past with face-to-face team projects to online classes were, at best, disappointing. I assigned a project to my online courses that required student groups of three or four to divide work among themselves and to integrate their results to make a final product. I set virtual meeting times for each of the student groups, and unfortunately, the student groups did not meet during the scheduled times because of scheduling conflicts. I then used email and discussion posts to facilitate student group meetings and noticed those communication methods proved to be problematic as well; there were communication delays between students, students failed to assume project work responsibilities, and some students exerted less effort than other students. The common factor that I attribute to the issues listed above is that there were too many students in each online group.

New Approach

When I attend technical software meetings and talk to employers, graduates, and other local software developers, I hear a lot about "working remotely." Clearly, online collaboration and communication are more important than ever before; many jobs in the modern workplace require online collaboration. Knowing the importance of collaboration, as well as understanding how crucial it is to learning, increased my resolve to find an alternative, more workable approach for online student group work.

I abandoned assigning online course projects to large student teams, and instead, I now assign projects that only require two students to collaborate with each other. I found that as I narrowed the online student groups down to pairs, many of the problems exhibited in the larger student groups disappeared.

Collaborative Project With Pairs

In my software development class, each student is required to create a short scenario and post it to the assigned discussion thread in the learning management system (LMS). Scenarios are just everyday problems that require a decision or imaginative outlines of simple games or simulations. I set ground rules for scenario construction so the scenarios are complex enough to incorporate key programming concepts, but simple enough to be completed in a week or two.

For example, consider the steps that students Alicia and Brandon take:

- Alicia poses a scenario with an initial post on the discussion thread in the LMS.
- Brandon chooses Alicia's scenario from the discussion board and claims it with a reply to Alicia's post.
- Brandon designs and codes the scenario in the appropriate programming language. He posts his code as a reply to the thread.
- Alicia, the scenario creator, reviews and tests Brandon's code and comments on his implementation with another reply.

The project sets up a structured exchange between two students who solve a problem together.

Project Example

Alicia might post: A coffee shop, Galaxybucks, has four sizes of cups: Small, medium, large, and extra large. The coffee cup prices are $1, $2, $3, and $4, respectively. The shop also has toppings for the coffee drinks. Customers can choose from whipped cream, caramel, and hazelnut for an extra 50 cents each, and chocolate for an extra 75 cents. If a customer purchases two medium or larger drinks, they receive 10 percent off of the total price. Joe has $10 and wants to buy one medium coffee with a whipped cream topping and one extra-large coffee with a double chocolate topping. Does Joe have enough money to purchase the coffee drinks? How much change will Joe have after purchasing the coffee drinks?

Brandon claims the scenario with a post stating, "claimed." Two students may claim the same scenario, but they must eventually compare and contrast their implementation of the scenario with the other student, which ensures there are enough scenarios for all students.

Brandon analyzes the scenario and designs code to implement it. During this step of the project students attempting the scenario ask the student who created the scenario for clarification. A possible question Brandon might ask Alicia is: "What should the outcome be if the customer only buys one extra-large coffee?" Ultimately, Brandon posts his code and test results to the thread for Alicia to read. Alicia then reviews Brandon's code, performs some further testing, and adds helpful comments.

The student posting his or her code to the thread appreciates feedback from the scenario creator on his or her coding efforts. Scenario creators are also flattered when a peer does a good job with their scenarios.

The Outcome

I've used this collaborative programming technique in dozens of sections across multiple courses and the results are rarely boring. In fact, the results are usually entertaining, engaging, and educational. It's engaging for students because it offers choices rather than prescribed exercises, it is an outlet for their creativity, and it provides a structured framework for interacting with at least one other classmate. It is educational because student peers are excellent teachers when instructors make project circumstances compelling.

Other advantages include:

- By implementing another student's unique ideas, the potential problem of students just copying and pasting from the internet is reduced.
- All students in the class can see their peers' codes, yet there is little opportunity for plagiarism, since the scenarios are almost always different.

Conclusion

The pair-based approach increases engagement in an online environment that otherwise may seem very sterile. Placing students into pairs for a project can be applied across various disciplines and courses. Decreasing student group numbers down to pairs reduces scheduling conflicts and confusion among students. There are certainly costs to pair-based projects and it doesn't always work for every student. This pair-based approach assumes that your institution's LMS supports threaded discussions. Also, instructors must actively monitor discussions to provide feedback, help strengthen weak scenarios, and troubleshoot any problems.

Jerry Reed, *Professor, Software Development*

For further information, contact the author at Valencia College at 1800 South Kirkman Road, Orlando, FL 32811. Email: greed9@valenciacollege.edu

June 21, 2018, XL, No. 21
©The University of Texas at Austin, 2018
www.nisod.org |

Opinions and views expressed are those of the author(s) and do not necessarily reflect those of NISOD. *Innovation Abstracts* is published weekly during the fall and spring terms of the academic calendar, except Thanksgiving week, by NISOD, College of Education, 1912 Speedway, D5600, Austin, Texas 78712-1607, (512) 471-7545, Email: abstracts@nisod.org