MULTI-PHASE HOMEWORK ASSIGNMENTS IN CS I AND CS II*

PANEL DISCUSSION

Jim Huggins
Computer Science Program
Kettering University
1700 West Third Avenue, Flint, MI 48504
810-762-9500 x5439
jhuggins@kettering.edu

Clif Kussmaul
Mathematical Sciences Department
Muhlenberg College
2400 West Chew St, Allentown, PA 18104
484-664-3299
kussmaul@mathcs.muhlenberg.edu

Amruth Kumar
Computer Science Department
Ramapo College of New Jersey
505 Ramapo Valley Road, Mahwah, NJ 07430-1680
201-684-7712
amruth@ramapo.edu

John A. Trono
Computer Science Department
Saint Michael's College
One Winooski Park, Colchester, VT 05439
802-654-2432
jtrono@smcvt.edu

ABSTRACT

All of the panelists have used small sets of related programming assignments in introductory CS courses. These assignments are essentially larger programs which are developed during several separate phases. This approach has several advantages:

- Students are able to develop more realistic and interesting programs.
- Students are motivated to write better code as well as documentation. Those who don't are quickly confronted by the implications.

- Student interest in completing the projects is better sustained, since there is continuity from one project to the next.
- It models desirable techniques such as iterative development and encapsulation.
- It demonstrates some of the challenges inherent in modifying and maintaining code in response to evolving requirements.
- Anecdotally, it seems that more students successfully complete the resulting program than when treating it as one large assignment.

There are also some disadvantages:

- The penalty for not completing a multi-phase assignment is generally harsher.
- Projects may not exercise the student's problem-solving skills as well as assigning several unrelated projects, each of which must be solved ground-up.

Each panelist will enumerate their own specific reasons for using this approach, discuss specific assignments as examples, and describe the advantages and disadvantages they have experienced. We will leave sufficient time afterwards for discussion of this strategy.

JIM HUGGINS - KETTERING UNIVERSITY

We use a multi-phase project in our CS II Java course. The project involves multiple revisions of a simple database program; each revision involves replacing the data structure used in the previous version. The final revision involves adding a Java GUI to the program. The final product (in most cases) is of substantial size and functionality.

The project has proven to be beneficial in several ways. Pedagogically, it reinforces the value of modular design, as students are forced to discard and rewrite code multiple times. It also allows students to focus on the core topics (i.e. data structures) while minimizing the time spent on infrastructural issues (e.g. data file formats, console I/O requirements). For the instructor, the project can be easily varied to allow for different projects without changing the essential nature of the project, making it easier to administer.

AMRUTH KUMAR - RAMAPO COLLEGE OF NEW JERSEY

We have been using multi-phase projects in CS I/II since Spring 97. We have observed that:

- Multi-phase projects provide a gentler ramp for weaker students to get up to speed when doing their assignments. Hence, they are especially helpful in retaining the weaker students and improving their problem-solving skills.
- Students tend to get serious with the projects earlier in the semester. Students who do not keep up with the work are under no illusions about their progress in the course, and are likely to drop the course rather than stay till the bitter end.

Typically, our CS I projects involve comparing among and computing with multiple alternatives for which data is available online [1], such as converting among currencies, comparing long-distance telephone providers, selecting among many stocks, choosing among different cars, and telling fortune based on zodiac signs. Our CS II projects involve billing

problems, such as for student tuition, long-distance telephone, credit card statement, automobile insurance policy and payroll.

CLIF KUSSMAUL - MUHLENBERG COLLEGE

I have been using multi-phase assignments primarily to help students develop better habits to prepare them for larger assignments in subsequent courses. Most recently, I have become interested in using these assignments to encourage test-driven development (TDD). In TDD, unit tests are written before the actual code is implemented. The tests help guide the design process, and provide a measure of how well the code works, and whether anything stops working as a result of subsequent changes. Usually, the unit tests are significantly simpler to write and debug than the code being tested.

Thus, in the initial assignment(s), students develop test code. In some cases, they are given a black-box implementation to validate their tests. In subsequent assignments, they develop code to implement their tests. They can run the tests to verify that new functionality works correctly, and that existing functionality has not been compromised.

JOHN A. TRONO - SAINT MICHAEL'S COLLEGE

We have used this style of assignment in our beginning programming courses since the early 1990s. Even though each of the assignments will create a viable, useful piece of software, the final program is just one goal of this approach. The major benefit of such a strategy is that students experience the inherent problems in modifying and maintaining software as well as handling the problems that arise when requirements evolve over time.

When used in a CS I course, this type of assignment can be thought of as extending, or enhancing, a previous version of some software artifact. In CS II, this approach can aid in reinforcing important concepts like abstraction, data encapsulation, and information hiding. By allowing the students to complete a straightforward implementation of the program, subsequent assignments can require that several modules be replaced with equivalent - but improved - code. (These typically emphasize the creation of a more efficient implementation.)

[1] Kumar, A.N., "Detecting and Preventing Plagiarism in Projects", *The Journal of Computing in Small Colleges*, Vol 13(5), May 98, pp 132-138.