

The case for not doing design in the first year

Claire Bono
University of Southern California

Intro students can't design software. In the procedure-oriented paradigm, many can't figure out how to generalize a procedure so it can be reusable within their application. In the object-oriented paradigm they can't figure out how to partition their problem into objects/classes well, can't design a good interface to a class, and can't figure out when inheritance would be appropriate. They can't figure out when it's appropriate to use which data structures.

How to turn these negatives into a positive? A post-CS2 design course. Once students have mastered the low-level skills in the first year, they will be ready to get more practice designing and building larger pieces of software, using the techniques and tools necessary to do it effectively. Such a course has some things in common with software engineering courses, but focuses exclusively on the software-building, rather than economics or management aspects. Perhaps we should call it programming-in-the-large. Some possible activities and topics for such a course:

- continue (from earlier courses) showing students examples of good designs
- students do bigger programming assignments in small teams, with a significant portion of the grade based on style / design concerns
- design methodology, such as responsibility-driven design (CRC cards)
- design notation, such as UML
- design patterns
- principles of GUI design
- use of a class library and/or application framework
- how to choose data structures / algorithms appropriately (as opposed to how to implement -- goes along with use of library such as STL)
- debugging and testing techniques such as program profiling, test scaffolding and stubs

What is the impact on CS 1 and CS 2 if you offer such a course? Rather than expecting students to create good designs, CS 1 and CS 2 instructors can show students examples of good designs. By taking design instruction out of CS 1 and CS 2, these courses have more time to focus on programming-in-the-small skills. E.g., writing correct loops in CS 1 and getting practice with the low-level details of pointer structures in CS 2. Students can practice these skills in the context of object-oriented programs, e.g., using pre-built classes, overriding functions from base classes, and/or implementing a class specified by the instructor.

I teach a version of such a design course at USC. Such courses are also offered at Duke and Stanford, as well as other universities.