

Stephen Wong, Oberlin College Position Paper

In the 1999-2000 academic year at Oberlin College, I experimented with a fairly radical “objects-with-design-patterns” approach in our CS1/CS2 course. The goal was to see what and how much difference such an approach would have with regards to the more traditional format used in the past. We stayed with Java as the language of choice but emphasized OO design right from the start.

As early as the second week of classes, I introduced inheritance/polymorphism coupled with composition as indispensable and foundational elements of OOP. Design patterns are used as concrete examples of how these concepts are put to use in elegant and effective ways. Design patterns proved to be an extremely useful and effective tool to both teach students sound OO design and analysis, and as a means of providing a tangible structure and standardized language for abstract concepts. To illustrate the design patterns and to help students focus at the desired level of abstraction, I extensively used UML, CASE, and RAD tools. These tools constituted an integral part of my teaching.

Overall, the experiment was a success. The students accomplished significantly more than in previous years and were writing significantly more sophisticated programs with much better underlying architectures. Most importantly, the students learned to solve problems by structuring solutions through object modeling rather than through brute force coding. The students were also able to create “real” programs with GUI’s and capabilities similar to what they see in commercial programs. This helped drive home the utility and importance of the concepts and techniques they were learning.

As it was the first time I had tried these approaches, the course was not without its rough edges. It was a very difficult course to teach, requiring a rethinking of every aspect of the lectures and laboratories. Everything was redesigned and rewritten from the ground up. The learning curve the students went through was different than that in our traditional course and will take some time to fully understand and accommodate. For instance, students find syntax to be reassuring and a design-driven approach, with its de-emphasis on syntax, can cause non-trivial amounts of anxiety. This anxiety “hump” is unnerving, but was overcome by most students by about two thirds of the way through the first semester. The lectures and laboratories were closely coupled to and relied extensively on the adopted the CASE and RAD software. The course entailed much more student-teacher interaction and close monitoring of students’ work.

On the other hand, we were rewarded with some of the most advanced CS1/2 graduates we have ever produced. For instance, in their final projects, which the students designed and implemented from scratch, we saw such things as multi-threaded arcade games, virtual token ring (one was self-repairing!) and star networks using RMI, dynamic behavior modification of objects, and abstract command (message) passing. Several students have recently communicated that their summer employers have been very impressed with their knowledge and skill levels, particularly in the areas of OOP and design patterns and that the courses helped them land those jobs.