Breadth with Depth in a CS1 Course

Thomas Murtagh
Williams College

Over the last 8 years, I have been involved in the design of two new CS 1 courses. In 1999, our department transitioned from a programming-focused CS 1 course based on Pascal to a course using Java. In the process, we completely redesigned the course with two goals in mind. We wanted to fully embrace the object-oriented features provided by Java. We also wanted to exploit the libraries included with Java to enable our students to construct programs that implemented interesting functionality and modern interfaces. I collaborated with two other members of our department, Kim Bruce and Andrea Danyluk, on this project. In addition to the new course, our efforts led to a Java library named objectdraw [3] and a text entitled *Java: An Eventful Approach* [2].

While working on this course, I became convinced that a programming-focused course was not the best way to introduce students to computer science. I feared that many students left our course thinking computer science was nothing but programming. It seemed clear that some form of breadth-first introduction to the discipline would be better, but existing breadth-first approaches seemed to have a major flaw. They lacked coherence. As the authors of Computing Curriculum 2001 observed:

> The many disparate topics typically found in a breadth-first course must be tied together into an integrated whole. Students must not see the course as a collection of interesting but unrelated topics in a "if this is Tuesday it must be computer organization" style [1].

With these concerns in mind, I proposed a new approach to CS 1 designed to accomplish many of the goals of breadth-first courses while tightly integrating the topics covered [5]. The key to my approach is to interpret the notion of "breadth" in a somewhat unusual way. In most breadth-first curricular designs, "breadth" has been taken to imply the coverage of many different subfields of computer science [8, 9, 10, 11]. I believe that it is unnecessary and inappropriate to introduce students to such a large collection of subfields. The differences between subfields of computer science is not what we should emphasize. Instead, we should be presenting the common elements that interconnect subfields of computer science [4]. The real goal should be to show students the types of problems we address, the techniques we use to attack these problems, and the forms taken by the solutions we devise.

In the extreme, the exploration of any single subfield of computer science should provide the opportunity to expose students to these defining aspects of our discipline. I proposed adopting this extreme. I explored ways to implement this approach for several years. Then, in the fall of 2005, our department instituted such a course as our first course for majors. We are now offering the course for the fourth time.

This new course is focused on the topic of digital communications and computer networks [6]. Networking topics covered include techniques for transmitting binary data, data compression schemes, broadcast networks, routing algorithms, basics of TCP/IP, and codes for error detection and correction. These topics provide students with a reasonably complete understanding of how computer networks operate. At the same time, they provide an integrated tour of many topics that would generally appear in a "Great Ideas in Computer Science" course. It is my hope to develop similar courses using topics other than networking as the integrating thread in the future.

An introductory course that does not impart significant programming skills would necessitate lengthening the chain of courses entering majors must complete. Our course therefore presents an introduction to programming that covers at least 75% of the material presented in the course it replaced, including standard topics such as conditionals, loops, classes, methods, recursion, arrays, and strings. Our presentation of programming is tightly integrated with the topic of networking that is used to motivate other material presented to the students. The programming examples and laboratory assignments explore algorithms and applications associated with the Internet.

The process of designing and teaching both the course associated with *Java: An Eventful Approach* and our new networking-focused CS 1 course has led me to explore many of the seven issues listed in the "Rationale" for this workshop.

1 & 2) I feel strongly that the goal of CS 1 should be to enable students to understand the nature of computer science. This will better serve potential majors by enabling them to make well-informed decisions both about whether to major and how to select courses within the major. A course focused on a single field within computer science can provide a way to both accomplish this goal and to serve non-majors. By selecting an appropriate topic as the unifying theme for the course, we can provide non-majors with knowledge that is more useful to them than programming skills.

3, 4, & 5) In the design of both courses, we have struggled to find ways to introduce object-oriented programming early in an effective way. In both courses we have also sought to exploit relatively advanced features of the Java libraries (GUI components, graphics primitives, and networking primitives) to challenge and inspire our students with interesting examples and assignments. This has led to the development of two support libraries named objectdraw and Squint. Interestingly, although these libraries are quite different in many ways, they support a common approach to introducing object-oriented programming [7].

7) A course designed to enable students to understand the nature of computer science must provide students with some sense of the role mathematics plays in our field. In our current course, we accomplish this goal by exploring the application of discrete probabilities to several problems in networking.

References

[1] ACM/IEEE Joint Task Force on Computing Curricula. *Computing Curricula 2001*, volume 18, 2001.

[2] K. Bruce, A. Danyluk, and T. Murtagh. *Java: An Eventful Approach.* Prentice-Hall, Upper Saddle River, New Jersey, 1st edition, 2006.

[3] K. B. Bruce, A. P. Danyluk, and T. P. Murtagh. A library to support a graphics-based objects-first approach to CS 1. In *SIGCSE '01: Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*, pages 6–10, 2001.

[4] P. J. Denning. Great principles of computing. *Commun. ACM*, 46(11):15–20, 2003.

[5] T. P. Murtagh. Teacing breadth-first depth-first. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*, pages 37–40. ACM Press, 2001.

[6] T. P. Murtagh. Weaving CS into CS1: A doubly depth-first approach. In *SIGCSE '07: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, pages 336–340, 2007.

[7] T. P. Murtagh. Squint: Barely visible library support for CS1. In *SIGCSE '07: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, pages 526–530, SIGCSE '07: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education.

[8] A. T. Phillips, D. E. Stevenson, and M. R. Wick. Implementing CC2001: A breadth-first introductory course for a just-in-time curriculum design. In *SIGCSE '03: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, pages 238–242, 2003.

[9] K. D. Powers. Breadth-also: a rationale and implementation. In *SIGCSE '03: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, pages 243–247, 2003.

[10] C. Shannon. Another breadth-first approach to CS1 using Python. In *SIGCSE '03: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, pages 248–251, 2003.

[11] A. Tucker and D. Garnish. A breadth-first introductory curriculum in computing. *Computer Science Education*, 32:271–295, 1991.