A Breadth-First Companion for the CS I Course

John E. Howland, Mark Lewis,
Thomas Hicks, and Gerald Pitts
Department of Computer Science
Trinity University
715 Stadium Drive
San Antonio, Texas 78212-7200
Voice: (210) 999-7380
Fax: (210) 999-7477
E-mail: jhowland@Trinity.Edu, mlewis@Trinity.Edu
E-mail: thicks@Trinity.Edu, gpitts@Trinity.Edu

April 14, 2003

Abstract

A breadth-first course, designed to supplement the CS I course, is described. The rationale for the course is given and early experience with the course is described. [1]

Subject Areas: Computer Science Education, Computer Science Curriculum.
Keywords: Breadth-first CS I course.

1 Introduction

The Trinity University Computer Science Department periodically reviews its curriculum. The Department receives input from student exit interviews, an industry advisory board, an alumni advisory board, student course evaluations and current computing curriculum reports. In a previous curriculum review Howland reported [How 1999] on a sequence of design seminars where second and third year majors were given progressively more difficult design problems culminating in a fourth-year cap-stone design project.

In our most recent review, we focused on needed changes in our CS I, II, and III courses. The choice of programming language used in these courses is a source of on-going debate within the computer science education community. Announced changes in the AP testing program indicate a move toward the use of Java within secondary computer science programs. Also, *Computing Curricula 2001 Computer Science* [Com 2001] report technological and cultural changes to which our curriculum must respond.

[1] This paper was published in the Journal of Computing Sciences in Colleges, Volume 18, Number 4, Pages 11-15, April 2003. Copyright ©2003 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

2 Computer Science Curriculum Reports

Chapter 6 of [Com 2001] gives several models for handling the first two or three courses. These include:

- *Imperative first*
- *Objects first*
- *Functional first*
- *Breadth first*
- *Algorithms first*
- *Hardware first*

Until recently, our introductory curriculum utilized the *Algorithms first* approach using C++ gradually increasing its use of object orientation during the first three courses. Because of changes in the AP testing program and the inevitable use of Java in high schools our department felt the need to move towards a CS I course in which students would focus mainly on programming using the C programming language. This change brings us more closely in alignment with the *Imperative first* model. Some may view this change as a step backwards, however, we feel that this change provides a leveling experience for our students who come to the program with only Java programming experience. C requires a non-object oriented contact with computing hardware and our course also uses the Unix operating system. The CS II course continues an algorithms focus using Java and introduces an object orientation in algorithm design. The CS III course is focused on the principles of data abstraction and uses the C++ language. Use of three languages during the first three courses helps enforce the multilingual cultural shift within the computing discipline, but still provides some depth of instruction during these courses because of the relationship of C and C++ and the object orientation of Java and C++.

3 Breadth-First Courses

"Computing Curricula 2001" [Com 2001] argues that because computer science is an ever-expanding field that includes many activities beyond programming, algorithms, and design, that there is an increasing need for courses which give a glimpse of the entire discipline.

"Computing Curricula 1991" [Tuck 1991] and the earlier "Computing as a Discipline" [Denn 1989] made a very strong case for introductory courses which not only provide instruction in programming and algorithms, but also introduce material from all the other sub-disciplines as well. Changes within computing science since the time of these reports have made the need for breadth-first input even greater. We have decided to address this problem by offering a supplementary breadth-first seminar course for first-year students which provides introductory experiences with each of the major sub-disciplines of the computing field.

4 The Breadth-First Supplement Course

One disadvantage of the major focus on algorithms and programming skills during the first three courses is that depth in these areas comes at the expense of knowledge of the breadth of the computing field early in the computer science major.

Our computer science curriculum immerses students in an intense programming environment. Students spend the first three semesters writing large amounts of code with little vision of the vastness and diversity of the computer science field. Since students generally have no conception of topics such as computer graphics,

artificial intelligence, theory, networking, database, etc., it is difficult, if not impossible, to anticipate future areas of concentration and research. It is equally difficult to determine those courses that students might need to take to optimize their efforts in those unknown areas. The goals of the breadth-first course include:

1. to provide an early overview of computer science
2. to establish early areas of interest
3. to better select electives and support courses to facilitate those interests
4. to encourage earlier participation in research projects
5. to provide better associations and relationships within individual courses
6. to improve student retention by providing an early look at non-programming topics
7. to make sure that students don't get to their senior year before they find sub-discipline of computer science that really turns them on

To deal with this problem we have designed a breadth first 1 credit seminar course to be taken at the same time a student enrolls in CS I. The seminar is titled "Major Ideas in Computer Science" and is team-taught by the entire computer science department faculty. Each faculty member gives two or three introductory lectures (usually on a topic which is closely related to their research interest) above and beyond their regular teaching load. We have not found this to be a problem as the effort to provide elaborate preparation for these lectures is re-used with little additional effort in subsequent offerings of the course. An example of the type of materials prepared for a topic may be found in [How 2002].

Topics covered include:

- Database Systems
- Functional Programming
- Artificial Intelligence
- Operating Systems
- Computability
- Computer Graphics
- Human Computer Interface
- Parallel Processing
- Programming Languages
- Numerical Algorithms
- Formal Methods
- Computer Networks
- Computer Architecture
- Simulation and Modeling
- Computer Gaming
- Robotics
- Genetic Programming
- Web Site Design

Attendance is required at each lecture and some reports on certain lectures are required, but the primary component of the student's grade is determined by attendance.

Students are exposed to the breadth of the computing discipline without giving any significant depth in any of the above topics. The idea is to provide a glimpse of the interesting parts of computing to offset some of the drudgery of an intensive depth-first look at algorithms and programming in the first three CS courses.

5 Conclusions and Recommendations

Experience from the first offering of the course has been encouraging. We are committed to improving the content and quality of course materials being developed for the course. One planned change is to provide an in-class laboratory component for some of the topics and report writing on other topics. This change is in response to complaints about a grading system which is based primarily on class attendance. We found that almost all students felt that the course expanded their overall view of computer science as being much more than just programming. A majority of students felt they had a better idea of areas of future concentration or research and nearly all felt they had learned about a few new areas of computer science.

Some representative comments from students enrolled in the course include:

> "I like the concept of being exposed to lots of different computer science topics."
> "It familiarizes us with the department faculty; it makes us feel like part of the group."
> "Pretty cool to see how all the different faculty teach."

Not all topics are equally interesting to all students. One student commented

> "I was dreading the Theory component, but I really liked it."

The Major Ideas of Computer Science Seminar, because it is a single course which is taken by all of our first-year students, provides a sense of community amongst our first-year students.

It appears that our students are developing a better understanding of Computer Science as a discipline. Some first-year students are indicating an interest in research topics and are attempting to be involved in research projects at a much earlier time than in the past. We do not have enough experience yet to determine the effects on student retention in CS I, II, and III, but we anticipate a positive effect as students see, at a much earlier point in their student careers, the possible outcomes of their efforts in the introductory course sequence.

References

[Com 2001] *Computing Curricula 2001 Computer Science, Final Report*, The Joint Task Force on Computing Curricula, IEEE Computer Society and Association for Computing Machinery, IEEE Computer Society, December 2001.

[Denn 1989] Denning, Peter J., Comer, Douglas E., Gries, David, Mulder, Michael C., Tucker, Alan B., Turner, Joe, and Young, Paul R., "Computing as a Discipline", Communications of the ACM, 32(1):9-23, January 1989.

[How 1999] Howland, John E., Konstam, Aaron, and Pitts, Gerald, "Focusing on Design", Journal for Computing in Small Colleges, Volume 14, Number 4, March 1999.

[How 2002] Howland, John E., "Functional Programming and the J Programming Language", http://www.cs.trinity.edu/~jhowland/math-talk/functional1/, 2002.

[Tuck 1991] Tucker, Allen B., Barnes, Bruce H., Aiken, Robert M., Barker, Keith, Bruce, Kim B., Cain, J. Thomas, Conry, Susan E., Engel, Gerald L., Epstein, Richard G., Lidtke, Doris K., Mulder, Michael C., Rogers, Jean B., Spafford, Eugene H., and Turner, A. Joe, *Computing Curricula '91*, Association for Computing Machinery and the Computer Society of the IEEE, 1991.