Teaching an Integrated First-Year Computing Curriculum: Lessons Learned [‡]

D. Cordes, A. Parrish, B. Dixon, R. Pimmel, J. Jackson, R. Borie
University of Alabama

Abstract: *This paper describes an integrated first year curriculum in computing for Computer Science and Computer Engineering students at the University of Alabama. The curriculum is built around the basic thrusts of the Foundation Coalition, and provides an interdisciplinary introduction to the study of computing for both majors.*

Introduction

The University of Alabama is one of seven schools (Arizona State University, Maricopa Community College District, Rose-Hulman Institute of Technology, Texas A&M University, Texas A&M University – Kingsville, Texas Woman's University, and the University of Alabama) that are participating in the Foundation Coalition (FC), an NSF-sponsored undergraduate engineering education reform initiative. As part of this program, the College of Engineering has developed a new curriculum for freshman engineering. Using the FC's four basic thrusts (curriculum integration, active learning and teaming, technology-enabled education, and continuous assessment and evaluation), the College has put into place an entirely new freshman experience. Initial assessment results indicate that this curriculum has significantly higher retention rates than our traditional first-year program.

However, the revised curriculum is designed for traditional engineering majors. Students interested in either computer engineering or computer science were not seen (originally) as part of this curriculum's target audience. It was felt that students interested in the discipline of computing should instead focus on mastering fundamental computer literacy during their freshman year. This includes competence in programming, an idea of the internal operations of the machine (including data representation), and understanding of the various hardware components associated with a machine, and an appreciation for the fundamental concepts of discrete mathematics that provide a foundation for computing.

Nevertheless, as stated previously, the merits of a FC-based freshman experience are significant. Given this, the Department of Computer Science (CS) and the Electrical and Computer Engineering Department (ECE) requested (and received) funding from NSF to develop an integrated freshman year in computing.

Institutional Overview

The University of Alabama has approximately 400 undergraduates enrolled in either Computer Science or Computer Engineering, evenly split between the two majors. The

[‡] This work supported in part by NSF grants DUE-9652785 and EEC-9221460

two departments share a number of common courses. For example, the Electrical and Computer Engineering Department is responsible for all instruction in digital logic, assembly programming, machine organization, and computer architecture. Likewise, the Computer Science Department is responsible for the teaching object-oriented programming, data structures, software engineering, and operating systems. Approximately 28 hours of common courses are shared between the two majors.

Nevertheless, coordination of these two programs during the first year has not been optimal in the past. This was due, in part, to a difference in opinion regarding the programming language of choice for freshman students. The CS department preferred Ada, while the ECE department utilized the C language.

With the introduction of the Foundation Coalition's freshman curriculum in 1994, students had the opportunity to take an eleven-hour integrated sequence of mathematics, science, and engineering. The first-year of the FC curriculum is shown below.

Fall Semester		**Spring Semester**	
Integrated Calculus I	4	Integrated Calculus II	4
Integrated Chemistry I	4	Integrated Physics I	4
Introduction to Engineering I	3	Integrated Chemistry II	4
Freshman English I	3	Introduction to Engineering II	3
		Freshman English II	3

Students in ECE began participating in this curriculum immediately. However, due to the FC's lack of emphasis on programming in the first year, students in CS did not initially participate in this curriculum.

After a couple of offerings of the FC freshman curriculum, it became apparent that numerous benefits existed within this model. Retention rates for students were up approximately 20% over the traditional curriculum, and student interest in the FC curriculum was growing steadily. As a result, CS faculty began to work with FC leaders in an effort to bring the benefits of this new curriculum to their students. Specifically, two goals were realized:

a) Develop a FC-based curriculum with an emphasis on programming fundamentals
b) Determine if a common first year could be established for both CS & ECE majors.

The Integrated Curriculum

The resulting curriculum model retained the mathematics and basic engineering design concepts from the FC freshman curriculum, replacing the Chemistry courses with a sequence on programming. The curriculum model is shown below.

Fall Semester		**Spring Semester**	
Introduction to Computing I	5	Introduction to Computing II	5
Integrated Calculus I	4	Integrated Calculus II	4
Introduction to Engineering I	3	Integrated Physics I	4
Freshman English I	3	Freshman English II	3

The content included in the first-year introduction to computing was formally developed during the summer of 1997. The goal was to establish a course sequence that would, upon completion, provide the students with a solid foundation in: Programming, Digital logic, Discrete mathematics, Assembly language, and Machine organization.

The basic idea was to "weave" the instruction of these topics together into a coherent package. For example, when you introduce the basic data types (integers, doubles, etc.) to the student, you would also discuss how these values are represented internally within the machine (binary systems, two's complement notation, etc.). Linkages between computer software, computer hardware, and discrete mathematics were to be identified and exploited throughout the entire first-year curriculum.

This new course was team-taught, with one faculty member from CS (Allen Parrish) and one faculty member from ECE (Russ Pimmel) involved in its instruction. Brandon Dixon (CS) coordinated the laboratory exercises and programming assignments. It was offered for the first time in the Fall of 1997. Thirty-five students (21 from CS, 14 from ECE) were enrolled in this course. Of these 35 students, 24 were also participating in the FC freshman year experience.

Assessing the Curriculum

At the close of the Fall semester, the authors (course instructors) held a series of meetings to identify strengths and weaknesses of the integrated computing curriculum. These are identified below:

Apparent Benefits

1. Student motivation is increased. A marked difference was noted between interest in this course and traditional first-semester courses in either ECE or CS.
2. The quality of the programming assignments was significantly higher. The ability to develop problems from a specific (and appropriate) application domain was extremely beneficial.
3. Students benefited from the early introduction to computer hardware. The ability to provide a "hands-on" hardware experience to freshman proved enlightening to the students.

Perceived Weaknesses

1. Lack of emphasis on the discrete mathematics topics that provide the foundation for much of this material.
2. Lack of a continuous focus on programming throughout the semester seemed to hurt the student's software development skills.
3. Linkages between the various topics were not exploited to their fullest extent, primarily due to a lack of time within the course (given its current layout).
4. The time period between the student's introduction to digital logic and its application in the computer architecture course is lengthened.
5. Institutionalization of a monolithic five-hour course is difficult.

Looking at these weaknesses, it was recognized that three of these are directly content related. In addition to this input, we also received "electronic journal entries" from the FC students on a regular basis. Representative comments from these students regarding the integrated computing course are included below:

- *I have to work to get the grade, not to mention I'm learning something new and interesting ninety percent of the time.*
- *My best performance would have to be in CS, because I can grasp the concepts better, and it is something that I enjoy to do.*
- *It is interesting and I want to learn the material.*
- *My strongest area is my programming skills. In CS/ECE 131 we have learned many things about both hardware and software, but I've displayed an amazing aptitude for computer programming. I think this is because of my background in computers. I've always been around them, and I enjoy learning the technical aspects of them.*

Comments such as these confirmed our belief that the course we were providing was of benefit to the students. We had always believed in an integrated approach to computing. It appeared that the students also appreciated such an approach. Given this, we turned our attention to eliminating the various weaknesses we had noted in the initial offering of the course.

Lessons Learned

Specifically, we felt that the first offering of the course attempted to pack in too much material in too short of a time. Rather than attempt to teach programming and digital logic and relevant discrete mathematics in one semester, we felt that we should instead use the discrete mathematics to guide the introduction of other topics.

Specifically, we would focus more on fundamental issues from discrete mathematics in the first semester, and use this to introduce a number of topics in digital logic. The formal digital logic course would be shifted back one semester, but approximately one-third of it would be covered in the discrete mathematics course in the first semester. A revised outline for the first year is shown below.

First Semester
ECE 131: Discrete Mathematics: *boolean algebra, logic gates, K-maps, number systems and representation, combinatorics, state machines*
CS 131: Programming I: *sequence, selection, iteration, functions, arrays, I/O details*
Second Semester
ECE 131: Digital Logic & Hardware: *completion of digital logic course, introduction to basic hardware devices*
CS 131: Programming II: *classes, pointers, dynamic memory allocation*

In addition to the shifting of digital logic, we also decomposed the course into two separate co-requisite modules. The reasons for this were strictly administrative. Separate

course modules provides a friendlier environment for transfer and other non-traditional students, and also simplifies the staffing issue once this sequence is institutionalized.

Relationships exist between all four modules within the first year. Specifically, direct ties exist between the material in ECE I and II, where we first introduce the basic mathematical fundamentals associated with digital logic, then migrate into a complete course on digital logic and basic computer hardware. Likewise, linkages can be identified between fundamental issues in discrete mathematics and basic programming (representation of integers, truth tables and logical expressions, etc.). The programming courses obviously create strong linkages between themselves, and with the concurrent instruction of discrete mathematics and digital logic, we are able to provide the students with a problem domain that is both relevant and allows the construction of programs that are both interesting and challenging.

Summary

The authors are currently teaching the second semester of the integrated first year computing curriculum, and will also provide insights regarding its successes (and failures) at our ASEE presentation. In addition, we are currently revising our first-semester model to better address the issues identified above. These changes will be introduced into the course in the Fall of 1998.

We believe that we have established the basic framework of a workable model for the introduction of computing to freshman students. Given that the typical freshman is not necessarily aware of the differences in computer science versus computer engineering, this curriculum provides an excellent model for all students. Students taking this program of study can proceed into either computer science or computer engineering without any loss of credit.

Finally, we believe that by providing a unified approach to the discipline of computing, students are able to comprehend the “big picture” more easily. Rather than simply focus on the aspects of computing relevant to a single discipline, we provide a larger, broad-based approach to the entire discipline.