FIRST PROGRAMMING COURSE IN ENGINEERING: BALANCING TRADITION AND APPLICATION

K-Y. Daisy Fan, David I. Schwartz

Department of Computer Science
Cornell University, Ithaca, New York

ABSTRACT

The "Introduction to Programming" course is an essential part of any first-year engineering program. As part of a common first-year curriculum, one of the biggest challenges of this first programming course is *both* to teach fundamental programming concepts *and* give students practical tools that can be applied easily to upper level courses in different engineering disciplines. At Cornell University, we offer a new first-year programming course that uses two very different programming languages: MATLAB and Java. This one-semester course balances the need for working knowledge of a fundamental programming language, such as Java, and the need for working knowledge of an engineering computing tool so that upper-level courses can focus on high-level, conceptual issues rather than programming details. Furthermore, this new course accommodates students in arts and sciences who are interested in a programming course that has a strong mathematical focus instead of a more traditional programming course offered in a computer science department. This paper discusses the challenges in developing the course, the ambitious one-semester syllabus that teaches both MATLAB and Java with depth, our evaluation of this new course, and our plans for improving the programming course in a common first-year engineering curriculum.

INTRODUCTION

The "Introduction to Programming" course is an essential part of the first-year engineering curriculum that helps students develop expertise in some programming language, beyond spreadsheet computation. This first programming course is a service course, teaching computing skills that students will need in upper division engineering courses. Given the wide-ranging needs of different engineering disciplines, some undergraduate programs offer introductory programming courses within the different departments. Such a model allows the computing needs of the upper division courses in a specific field to dictate the syllabus of the introduction to programming course. This efficiency in the curriculum can incur a penalty for the students who change their major from one engineering field to another if they lack the prerequisite programming course for their new major.

At Cornell University, the College of Engineering has a "common first-year curriculum" philosophy that requires a common, one-semester, introductory programming experience for all students in the College. One of the biggest challenges is to offer a first programming course that not only teaches fundamental programming concepts, but also gives students practical tools that can be applied easily to upper division courses in different engineering disciplines. Teaching fundamental programming concepts is important because we want our students, our future engineers, to be able to mold software to suit problems rather to allow software to dictate limits. The Department of Computer Science, which offers this first programming course, has developed a new first-year programming course that uses two very different programming languages: MATLAB and

Java. This one-semester course balances the need for working knowledge of a fundamental programming language, such as Java, and the need for working knowledge of a powerful and popular engineering computing tool so that upper division courses can focus on high-level, conceptual issues rather than programming details.

This paper discusses the challenges in developing the course, the ambitious one-semester syllabus that teaches both MATLAB and Java with depth, our evaluation of this new course, and our plans for improving the programming course in a common first-year engineering curriculum.

CHALLENGES IN RE-DEVELOPING THE INTRODUCTORY PROGRAMMING COURSE

In recent years prior to 1999, the introductory programming course, CS100, was taught using Java, a fundamental programming language with object-oriented features. Students chose one of two tracks: one track relied heavily on mathematics, the other track did not focus on mathematics. In 1999, extensive debate and consultation among faculty in the College of Engineering revealed that faculty in the different departments were unsatisfied with how well CS100 prepared students for the computing needs of upper division courses. While knowledge of object-oriented programming was desirable in the field of computer science, most engineering departments showed increasing use of MATLAB, a computing environment, in their upper division courses and desired that more MATLAB be taught in CS100. Our engineering departments' increasing use of MATLAB for education is not unique. A number of universities now have introductory programming classes teach MATLAB formally, [1,2,3 (for example)] not leaving MATLAB to be introduced "on the fly" in upper division engineering courses. Equally clear from the consultation was the faculty support for the philosophy of the common first-year curriculum. Therefore, any changes made to CS100 had to be a compromise among the needs of all the engineering fields.

At Cornell, another consideration is that the Computer Science major is offered in two colleges: College of Engineering and College of Arts & Science. Any change in CS100 must accommodate students from both colleges. Specifically, although it is desirable for Calculus to be tied to CS100 for our engineering students, Calculus cannot be a pre- or co-requisite for the students from Arts & Science.

In order to meet the wide-ranging needs of the engineering departments and to accommodate both the Engineering and the Arts & Science colleges, two tracks of CS100 have been offered since the fall of 2000. The two tracks teach the same *programming concepts* but differ in the amount of focus on mathematics and on the amount of usage of MATLAB:

- CS100M involves seven weeks each of MATLAB and Java. This track requires some background in Calculus and has a strong focus on mathematics and engineering.

- CS100J is the traditional programming course based on 12 weeks of Java with a two-week introduction to MATLAB.

The same programming *concepts*, including control structures, program organization, algorithm development, and object-oriented programming, are covered in both tracks despite the difference on the amount of time spent on the two programming languages. *Either* track satisfies the introductory computing requirement in *all* the engineering fields. Students in Engineering and in Arts & Science choose between the two tracks freely depending on their interest and preparation in mathematics. The remainder of this paper presents the ambitious one-semester syllabus and our evaluation of this new course, CS100M.

CS100 Syllabi

CS100 is a one-semester course that introduces computer programming—no previous programming experience is assumed. During the 14-week semester, students learn how to write programs using MATLAB and Java, and most importantly, gain the skills and confidence needed to further develop their programming expertise in future course work or on their own.

The syllabi for the M and J tracks of CS100 are given in Table 1. The same programming concepts are taught in both courses using different amounts of MATLAB and Java content (MATLAB content is shaded in Table 1). CS100J is the "traditional" introductory programming course taught at Cornell based on an object-oriented programming language. Two weeks of MATLAB instruction augments this traditional course to give students a foundation in a computing tool that is widely used in upper division courses. CS 100M is the new course added in 2000 that uses MATLAB extensively but still keeps the tradition of teaching object-oriented programming in depth.

In CS100M, examples and exercises focus on mathematics and simple engineering applications. Fundamental programming concepts are taught using MATLAB, a language and a user-friendly computing environment.[4] As a result, students learn the introductory concepts quickly, and we can demonstrate powerful computing concepts early in the course to capture student interest. Furthermore, the user-friendly graphics capability in MATLAB allows students to visualize data easily, reinforcing the idea that computers, and programming in particular, are useful for scientific and engineering investigation. After seven weeks in CS100M, students have learned fundamental programming concepts already so object-oriented programming in Java, not basic programming techniques, is emphasized.

At the end of the semester, students from CS100M and CS100J are able to write MATLAB programs for simple engineering

Table 1. Syllabi for CS100M and CS100J

Week	CS100M	CS100J
1	Introduction to problem solving and algorithms, assignment, elementary functions	Introduction to problem solving and algorithms, assignment, input/output
2	Branching, scripts	Branching, input/output
3	Iteration	Iteration
4	1-dimensional array	Iteration
5	Functions, program organization	Classes, objects, methods
6	File input/out, strings, graphics	Classes, objects, methods
7	2-dimensional array, graphics	Classes, objects, methods, strings
8	Java fundamentals, branching	Strings
9	Iteration	1-dimensional array
10	Classes, objects, methods	Sorting, linear search
11	1-dimensional array, sorting	MATLAB 1-dimensional array, graphics
12	Classes, objects, methods	2-d arrays
13	Inheritance	MATLAB 2-dimensional array, functions
14	Strings, 2-dimensional array	Inheritance

applications and Java programs in an object-oriented style. Although CS100J students have less exposure to MATLAB, they build enough foundation to be able to use MATLAB in upper division engineering courses and to continue to develop their MATLAB expertise.

EVALUATION

There are three primary considerations to evaluate the effectiveness of integrating MATLAB in our introductory programming courses:

1. Has MATLAB motivated the students?
2. Do the students master the course objectives?
3. Do the CS100M students fare as well as the CS100J students?

To address the first point, CS100M has attracted just as many, if not more, students as CS100J since CS100M debuted in Fall 2001. Therefore, we can conclude that, at the least, MATLAB has not detracted a large population of students from taking CS100M.

To address the next two points, we should address how later courses build upon skills developed in CS100. Three of the nine undergraduate engineering majors offered at Cornell require a subsequent programming course, CS211: Computers and Programming. CS211 assumes a background of CS100 or equivalent experience and continues the development of algorithm analysis and object-oriented concepts, such as generic programming, graphical user interfaces (GUIs), and recursion, with about half of the course as overview of data structures. With only occasional use of MATLAB for plotting, CS211 primarily uses Java.

Given CS211's focus on advanced programming concepts within a Java framework and extension of CS100 concepts, some CS100 students worry about their eventual CS211 performance if they take CS100M. In fact, most students who expect to take CS211 choose the J track instead of the M track as the introductory course. Our reply to the students' concern is that both CS100J and CS100M cover the same concepts. However, until recently, we have not had enough data to test that assertion.

We have collected grade data from three semesters of CS211: Spring 2001, Fall 2001, and Spring 2002, ranging from C- to A+. The tables provide data, using three types of students:

- Students who took CS100J (312 students)
- Students who took CS100M (148 students)
- Students who did not take CS100 (258 students)

Note that students self-select their CS100 track—in CS211 there were twice as many students from CS100J than from CS100M. We excluded a negligible number of Ds and Fs, which amounted to fifteen students overall. Table 2 provides the specific numbers of students for each grade, and Table 3 summarizes the data for three overall grade categories of "A," "B," and "C."

Table 3 shows some disparity at the higher-grade levels. At Cornell, students may elect to place out of CS100 and start with CS211. Consequently, students with enough preparation and confidence to start with the second course tend to excel probably due to more than one semester of programming (or hacking) experience before taking CS211. Comparing CS100J and CS100M, it appears that a higher percentage of CS100J than CS100M students get "A" grades, but overall, taking CS100J or CS100M does not seem to help or hurt a student's overall performance.

We do not have an intermediate programming course that uses MATLAB. Upper division courses use MATLAB as a tool, not as a subject of study. Most engineering faculty are supportive of the new

Table 2. CS211 grades of students from CS100J, CS100M, and no CS100

	A+	A	A-	B+	B	B-	C+	C	C-
100J	10	61	46	36	62	33	20	36	8
100M	5	22	17	24	33	14	9	16	8
No 100	16	61	27	24	44	29	21	24	12

Table 3. CS211 grade distribution

	% of A	% of B	% of C
100J	37	42	21
100M	30	48	22
No 100	40	38	22

CS100M track as it teaches an important computing tool and highlights the usefulness of computing in engineering. However, some instructors have noticed a recent, and perhaps undesirable, disparity in MATLAB skills.

POTENTIAL CHANGES TO CS100M

CS100M is a relatively new course and its syllabus continues to evolve as a result of instructor and student feedback. Some instructors have noted that students find the transition from MATLAB to Java, a procedural language to an object-oriented one, difficult. To address this difficulty, we are considering altering the flow of the two languages.

In one scenario, we might break up the MATLAB and Java flow into three or more alternating modules. Such alternating modules might provide an opportunity to illustrate syntactical differences between the languages and develop proficiency in both. Moreover, patterns starting with Java and an "objects-first" approach might start students earlier on the more difficult concepts of object-oriented programming. A Java GUI environment, such as BlueJ, [5] would support such a venture for the Java portion of CS100M.

Another way to redistribute the MATLAB and Java material involves gradually mixing MATLAB with Java content part way into the semester. After about one quarter of the semester has progressed, students have developed enough programming experience to develop simple programs. By introducing relatively small amounts of Java based on the previous three weeks of material, students might not only reinforce early concepts, but become prepared for the more advanced concepts in Java later in the course.

An intriguing possibility that would enhance a variety of approaches is using MATLAB's integration of Java. A final project could be used to "merge" the content, so that students solve a more complex problem in ways for which the languages are suited.

CONCLUSIONS

Given the constraint of a “common first-year curriculum,” the Department of Computer Science at Cornell University has introduced a new course, CS100M, that teaches fundamental programming concepts as well as practical computing tools for the engineering community. CS100M introduces fundamental programming concepts using MATLAB, a popular computing tool, and teaches object-oriented programming using Java.

Inspection of student performance in the subsequent, Java-based, programming course CS211 shows that CS100M and CS100J students have done about the same, although more CS100J than CS100M students get “A” grades. However, the fact that CS100M students have been doing relatively the same as CS100J students implies that using a significant amount of MATLAB in an introductory programming course does not impede the development of fundamental programming skills. Measuring performance in CS211 gives only a rough guide. Further work needs to be done in refining our evaluation so that we can gauge the amount of MATLAB learned and perhaps adjust course content. For example, a similar analysis of grades could be done in other courses, especially to monitor possible disparity in MATLAB skills.

We expect to continue to improve the course as we gain more experience with the new CS100M course. The use of MATLAB and Java balances formalism and application, both of which are needed by the students. Learning about both practical computing tools and fundamental programming concepts will allow students to mold software to solve engineering problems, rather than be limited by software.

ACKNOWLEDGEMENTS

We gratefully acknowledge the gathering of the data used for the evaluation by our course administrator Laurie Buck. We also thank Professor Keshav Pingali, who initiated the evaluation, and Professor Charles Van Loan for providing much of the background information and invaluable insights. We thank all members of the Engineering College Computing Policy Committee (1999) for their consultation with and recommendation to the Engineering faculty.

REFERENCES

1. Freshman computing course, Purdue University College of Engineering. [Online]. Available: https://Engineering. Purdue.edu/Engr/Academics/Program/.fre .html

2. Freshman computing course, University of Pittsburgh School of Engineering. [Online]. Available:http://civeng1.civ. pitt.edu/%7Eeng12/

3. Freshman computing course, Rice University School of Engineering. [Online]. Available:http://www.cs.rice. edu/Undergrad/computing-overview. shtml

4. Fan, K-Y. Daisy and David I. Schwartz, "Introductory Programming Using MATLAB," MATLAB News & Notes, p. 4, Oct. 2002. [Online]. Available: http://www.mathworks.com/company/newsletter/oct02/programming.shtml

5. BlueJ. [Online]. Available:http://www. bluej.org/

BIOGRAPHICAL INFORMATION

Daisy Fan is an Assistant Professor in the Department of Computer Science at Cornell. She teaches computing and her area of research is optimization with application to environmental systems. She received her Ph.D. in Civil & Environmental Engineering at Cornell and her M.Sc. and B.Sc. in Civil Engineering at the University of Manitoba, Canada.

David Schwartz is an Assistant Professor in the Department of Computer Science at Cornell. He teaches computing and his area of research is educational technology. He has published two textbooks geared towards first-year engineering students. He received his B.S., M.S., and Ph.D. in Civil Engineering from State University of New York at Buffalo.