Back to Basics in CS1 and CS2

Stuart Reges
University of Washington
Computer Science & Engineering
Seattle, WA 98195
+206.695.9138

reges@cs.washington.edu

ABSTRACT

This paper describes a significant redesign of the introductory courses at the University of Washington that has led to increased enrollments, increased student satisfaction and an increase in the number of women admitted to the CS major. The new courses are still taught in Java, but they represent a return to the basics that were emphasized in the pre-Java era. The biggest changes have occurred in the CS1 course where we have replaced an "objects early" curriculum with a more traditional procedural approach using static methods in Java. The new CS1 course emphasizes problem solving, procedural decomposition and mastery of basic skills (e.g., loops, conditionals and arrays). The new CS2 course emphasizes data structures, linked lists, binary trees and recursion.

Categories and Subject Descriptors

K3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum.*

General Terms

None.

Keywords

CS1, CS2, objects first, problem solving.

1. INTRODUCTION

Curriculum 2001 defines the objects first approach as a course that "emphasizes the principles of object-oriented programming and design from the very beginning" and that "begins immediately with the notions of objects and inheritance" [13].

A few schools have completely reworked their CS1 course to fit this approach. Some have done so by developing custom libraries and toolkits, as with Williams College [3] and Northeastern University [11]. Others have taken advantage of custom development environments such as BlueJ developed specifically for CS1 [8].

Textbooks also are moving more in the direction of objects early. Cay Horstmann, for example, has moved his chapter "An Introduction to Objects and Classes" ahead of his chapter on "Fundamental Data Types" in his CS1 textbook [5] and Lewis and Loftus have moved their chapter on "Writing Classes" ahead of their chapters on control structures in their CS1 textbook [9]. This is reminiscent of the rush in the late 1980's to move procedures early in CS1 textbooks, documented in detail in a paper by Rich Pattis [10].

But not all schools experimenting with objects early have had a good experience. In late March, 2004, Elliot Koffman posted a message to the SIGCSE mailing list in which he said, "I fear we have reinvented the 'new math' syndrome and many of us are unaware of it." His message generated a flood of responses [2] that inspired several of us to organize a debate at the 2005 SIGCSE symposium about whether or not the objects early approach has failed [1].

I was thrust into the middle of this issue in the spring of 2004 when the faculty of the University of Washington hired me to take charge of their introductory courses and to "fix the problems." I had given a talk entitled "My Disillusionment with Java" in which I expressed my own concerns about the objects early approach and they hired me anyway.

The department was facing problems that are fairly common today in the wake of the dotcom collapse of 2001 and as a result of our collective struggle to figure out what to teach in intro courses and how to teach it. The major problems were:

- A decline in student satisfaction and enrollment in the department's introductory courses
- A decline in the number of applicants to the major, particularly among women
- Inconsistency in teaching as different instructors tried different approaches
- A lack of basic programming skills reported by instructors of upper-division courses

We solved the consistency problem by designating one individual (currently me) to take charge of the introductory courses and to design a standard curriculum for both the CS1 and CS2 courses.

We have now taught the new version of CS1 four times and the new version of CS2 twice. Early results indicate that we have addressed several of the other problems as well. Enrollments and student satisfaction are up as well as the number and percentage of women admitted to the CS major. It's too early to tell whether we have addressed the "basic skills" issue because these students haven't yet taken upper-division courses.

We made many changes all at once, so this does not constitute anything like a controlled experiment. We can't know for sure how much influence each change has had. But the bottom line is

SIGCSE'06, March 1–5, 2006, Houston, Texas, USA.

that we now have a stable set of introductory courses that are working well for the department.

2. THE NEW CS1

Our new version of CS1 looks a lot like a 1980's course taught in Pascal. We have gone back to procedural style programming. I was motivated to do this after attempting and failing to teach a broad range of introductory students at the University of Arizona using an "objects early" approach. I found that my best students did just fine in the new approach, but the broad range of mid-level students struggled with the object concept.

Like many instructors, I had tried several intermediate compromises. For example, I asked students to write what amounted to "procedural objects" where the methods of the object are similar to the input/process/output paradigm of procedural programming. This was unsatisfying because the resulting objects and programs were not very "object oriented." I also found that this led students to become dependent on what would be global variables in a procedural program. In other words, it was difficult to get them to understand when to declare instance variables and given that these were rather pathetic objects to begin with, it seemed unfair to take off style points for relying too heavily on "global" instance variables.

In the new version of the course we don't have students write their own classes until almost the end of the course. Instead, they decompose their programs into several static methods. This is essentially the same thing we did in Pascal, which means that we can return to many of the traditional issues that CS1 has addressed (decomposition, parameter passing, use of local variables, returning values from a method).

The syntax of static methods is cumbersome and serves as a constant reminder that Java was designed to be used in other ways, but this doesn't seem to bother the students. They quickly get used to typing "public static" in front of their methods.

Many instructors question the use of Java in a course that emphasizes procedural programming and I agree that Java is not the ideal language for this new version of the course. For us the payoff comes in the CS2 course where we can build on these basic skills. The same mid-level students who struggle with the object concept in an objects early course are likely to struggle with a change of programming language between CS1 and CS2. So even though Java is not an ideal choice for our CS1, we continue to use it because of its payoff in our CS2 course.

Our new CS1 course is not devoid of objects. Even as the students practice writing static methods, we have them use a number of interesting objects along the way, including a custom "drawing panel" object that allows us to have them do simple graphics. The key is that for most of the course they use objects without defining them. In the last two weeks of the class we discuss how to define objects and we end with a relatively simple program that allows them to practice defining their own classes.

2.1 Problem Solving

The first goal listed in the 1984 description of CS1 is "to introduce a disciplined approach to problem-solving methods and algorithm development" [7]. Numerous papers and textbooks over the years have referred to this essential aspect of CS1, that our course emphasizes "problem solving." In some sense this term is vacuous. Don't students in *all* of their courses solve some kind of problems? And yet we felt we knew what we meant when we used that phrase.

Donald Knuth gave some insight into this notion when he wrote that, "It has often been said that a person does not really understand something until after teaching it to someone else. Actually a person does not *really* understand something until after teaching it to a *computer*, i.e., expressing it as an algorithm" [6]. I believe Knuth has captured precisely what we mean by the term "problem solving."

Unfortunately this kind of problem solving has been disappearing as we have adopted the object oriented approach. We used to ask students to write complete programs that were specified at the program level (given these inputs, produce these outputs). That left a lot of problem solving for the students to work out. In our switch to objects we have more often given students highly constrained assignments in which we specify what classes they are to write and what methods to write within those classes. We are doing much of the problem solving for them.

I personally recommended this approach in a SIGCSE paper in which I said that I thought this would be good for the students [12]. It does give students experience writing to a specification and this is a good experience for students to have. The question is whether the experience of writing to a specification is more important than problem solving in the first course. I no longer believe the tradeoff is worth it.

Many instructors adopt a middle ground where they ask students to write methods that they know will require the students to introduce their own private methods. This retains some of the problem solving, but there is still a reduction in the complexity of the problems we are asking students to solve. I have also been surprised to find how much of a sense of satisfaction students derive from writing a complete program on their own.

Our switch to static methods has allowed us to bring back the problem solving aspects of the course that we thought were so important in the 1980's.

2.2 Basic Skills

In talking to faculty I have often heard the complaint that students seem to lack basic programming skills that they used to have. This is not a new complaint. It is almost a truism across universities and across disciplines to say that the senior faculty think that current students don't know as much as they used to. But there is good reason to believe that the new versions of CS1 we have been experimenting with do not give students as much practice as they used to get with basic programming skills.

All one has to do is count the number of lectures devoted to topics like conditionals, loops and arrays to see that we are putting less effort into the teaching of what we used to consider to be fundamental programming skills. Otherwise we wouldn't be able to have lectures on the details of writing classes, on inheritance and polymorphism, on the Java API (graphics, collections, etc) and on object oriented design. Assignments also focus less on basic skills because they have to include practice with these new programming constructs.

We have also found that when you write well-structured object-oriented code, you don't need the same set of skills. Event driven programs need fewer while loops than console and file based programs. Inheritance allows us to eliminate many of the conditional branches we so often include in procedural programs. And many of the array manipulations we used to have students write are now built-in operations in the collections classes.

Our decision to delay having students write their own classes has allowed us to return to assignments that force students to practice the traditional programming skills needed for procedural programming (decomposition, loops, if/else, arrays, etc).

2.3 Console and File Processing

For years Java has not provided a simple mechanism for console or file processing, but the Scanner class included in Java 5 finally provides a reasonable mechanism for both. Many people have argued that we should have students writing more modern programs with graphical user interfaces, but that introduces its own set of complexities.

Most of the programs that students write in our new CS1 course are console-based. This has allowed us to focus on other aspects of programming like decomposition and algorithmic thinking.

We also felt it important to put file processing back into our CS1 course. This opens up a wealth of programming assignments that we otherwise couldn't give. File processing is also a practical skill that students can use in other Java programs that they write.

2.4 Programming Assignments

Table 1 lists the programming assignments given in a recent offering of the new CS1 course. These are weekly assignments. The University of Washington is on the quarter system (approximately ten weeks per quarter).

Table 1. CS1 programming assignments

#	Description	Main Topics
1	display a song	println, static methods
2	display a figure (rocket ship)	for loops, nested loops, integer expressions
3	display a complex graphical image	value parameters, graphics
4	prompt for data to choose between two candidates for admission	interactive programs, if/else, return values
5	play a guessing game	while loops
6	display a graph of the popularity of a baby name	file processing, writing a mid-size program
7	analyze a file of responses to a personality test	file processing, arrays
8	define several "critter" classes as part of a simulation	defining classes

3. THE NEW CS2

The changes to CS2 have been less drastic because the course had not strayed too far from the traditional focus on data structures. But the switch to Java led the department to include additional topics like object-oriented modeling and graphical user interfaces. Over time these topics grew to such an extent that they were pushing out traditional topics like recursion, linked lists and binary trees.

In the redesign we gave up some of the OO material and all of the GUI material so that we could return to the traditional topics of linked lists, binary trees and recursion.

But we haven't abandoned OO issues. In the redesigned course we use the Java collections classes as a grand case study of how to use OO techniques like interfaces and abstract classes to efficiently implement a library of useful data structures.

3.1 Programming Assignments

Table 2 lists the programming assignments given in a recent offering of the new CS2 course. These are weekly programming assignments.

Table 2. CS2 programming assignments

#	Description	Main Topics
1	implement a sorted int list	arrays, defining classes
2	implement a "letter inventory" class	arrays, defining classes
3	manage information about a game of "assassin"	linked lists
4	implement the Sieve of Eratosthenes using a Queue	ADT's, using a queue
5	generate random sentences given a BNF grammar	recursion, use of Map, ArrayList, String.split
6	generate all anagrams of a phrase	recursive backtracking
7	use a binary tree to play "twenty questions"	binary trees
8	Huffman coding	binary trees, PriorityQueue

4. INDICATIONS OF SUCCESS

There are several indications that the new versions of the introductory courses are succeeding. Student evaluations have never been higher for the CS1 course. To get some historical perspective, we divided the recent student evaluation data into three groups. We have approximately five years of data from a C version of the course that was taught until 2001. We have approximately three years of data for the various Java versions of the course that were taught before the most recent change. And we have one year of data for the new Java version of CS1.

Table 3 shows mean course evaluation scores for these three versions of the course on the four questions that are meant to get an overall evaluation of the course and the instructor. These questions each had six possible answers and were converted into a numerical score that ranges from 0.0 to 5.0. Higher is better (0 corresponding to "very poor" and 5 corresponding to "excellent").

As the table indicates, average scores went down when the department first switched from C to Java, but the new version of the course is receiving higher student evaluations than either the old C version or the initial Java version.

Table 3. Overall course evaluation data

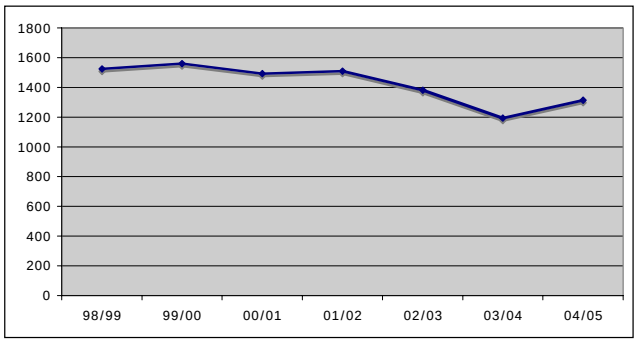
Question	C Mean	Old Java	New Java

		Mean	Mean
The course as a whole was:	3.71	3.30	4.06
The course content was:	3.71	3.40	4.01
The instructor's contribution to the course was:	3.82	3.42	4.43
The instructor's effectiveness in teaching the subject matter was:	3.70	3.20	4.35

We have only one quarter of data for the new version of the CS2 course, so it is difficult to draw definitive conclusions. But it is worth noting that the course received the highest score ever for the first question ("the course as a whole was").

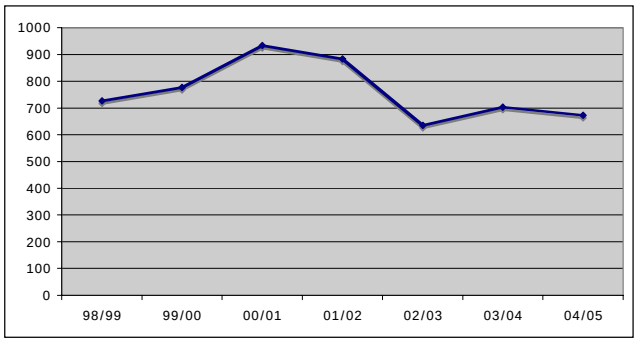
Enrollment has also turned around. We had seen a decline since the dotcom collapse in 2001. But with the new version of CS1, enrollment has stabilized and seems to be creeping back up. Figure 1 shows the annual enrollment in CS1 starting with the 1998/1999 academic year. Enrollments were 1500+ in the pre-dotcom years, but we had a fairly steady decline afterwards down to the low of just 1200 in the 2003/2004 academic year. Our department has not had the dramatic drop-off in enrollment that many schools have seen because it is required of many engineering majors and, therefore, has a base constituency. The last entry is for the new version of the course where we see a reversal of the downward trend (an increase of 120 students).

Figure 1. CS1 enrollment by academic year



We saw a more dramatic change in enrollment in our CS2 course after the dotcom collapse, as indicated in Figure 2. The high point occurred in 2000/2001 with a rapid decline in the subsequent two years. The enrollment has been fairly stable since then. The switch to the new version of CS1 does not seem to have changed the enrollment. The new version of CS2 has not been offered enough times to discern its impact.

Figure 2. CS2 enrollment by academic year



We have been particularly concerned about the percentage of women in our introductory courses and our major. Figure 3 shows the percentage of women in CS1 since 2000/2001. The graph shows a disturbing drop of nearly five percentage points from the peak in 2000/2001 to the low point in 2002/2003. The final entry is for the new version of CS1 and it indicates a significant increase in the percentage of women year over year (from 22.7% to 25.5%), although this is still not as high as the percentage in 2000/2001.

Figure 3. CS1 female enrollment by academic year

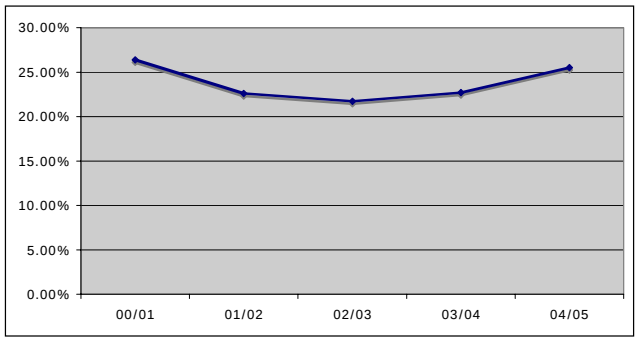
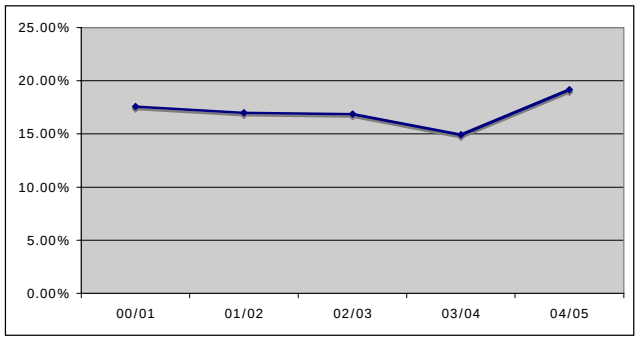


Figure 4 shows the corresponding data for the CS2 course. The percentage of women in CS2 has stayed fairly constant over time. We had a slight decline in 2003/2004 and a rebound in 2004/2005. We believe that this is due to unusual circumstances that might have artificially deflated the first number and artificially inflated the second. In any event, it is clear that the new versions of CS1 and CS2 are doing at least as well as the old versions in attracting women to the CS2 course.

Figure 4. CS2 female enrollment by academic year

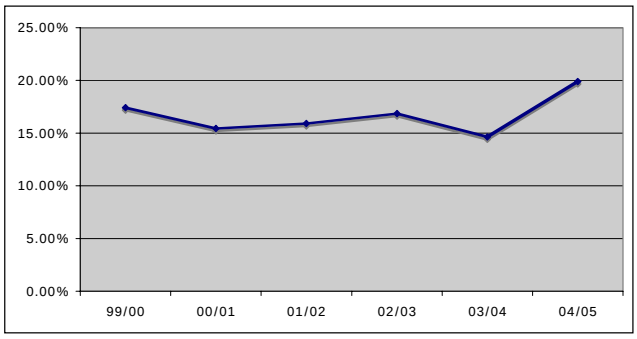


We won't know for a few years how the new versions of our introductory courses will affect admissions to the CS major, but we have some early indications that they are helping us to attract more women to the major. In the most recent round of admissions our pool consisted of students who had almost all taken the new version of CS1 and approximately half had taken the new version of CS2. We admitted 24 women out of 98 students (24.5%). This is the highest absolute number of women admitted to the program in one group and also the highest percentage of women admitted to the major. This is a very encouraging result, although we won't know for a while whether this represents an anomaly.

Figure 5 shows the trend in female admissions for the past 6 years. The admission rate for 2004/2005 is the highest we have

seen, although not quite as high as the 24.5% we saw in the most recent pool because it includes an earlier admissions group and a group of students who were admitted directly from high school.

Figure 5. Percent women admitted to CS by academic year



It is important to note that the improvements in female enrollment are a result of several efforts undertaken by the department to increase female participation. In particular, we have a seminar aimed at women in the intro courses that we believe is having a positive impact (described separately in [4]). We believe that the new versions of CS1 and CS2 are helping as well, but they are just part of the overall effort.

5. CONCLUSIONS

We have made many changes all at once, so it is difficult to determine which changes have produced which results. As the car companies are so fond of saying, "your mileage may vary."

What we can say definitively is that the changes we have made are working for us. So if anyone doubted whether a procedural emphasis in a Java CS1 course can work, they need doubt no more. We can serve as the existence proof that old-fashioned doesn't have to mean unpopular. We are teaching our courses to a broad range of engineering students who are more satisfied than they have ever been before with our course. At the same time, we are managing to retain a healthy percentage of women and to encourage many to pursue our major.

One specific worth noting is that we have accomplished this without the use of pair programming. In fact, we had been experimenting with pair programming in the previous version of the course. Our "back to basics" course requires students to write programs individually. Again, we don't know whether the switch to individual programming has made things better or worse or whether it's had no effect at all, but we can serve as an example of a school that is succeeding without pair programming.

6. REFERENCES

[1] Astrachan, A., Bruce, K., Koffman, E., Kölling, M. and Reges, S. *Resolved: Objects Early has Failed.* Proceedings of the thirty-sixth SIGCSE technical symposium on Computer Science, 2005.

[2] Bruce, K. *Controversy on How to Teach CS1: A discussion on the SIGCSE-members mailing list*. ACM SIGCSE Bulletin, Volume 37, Issue 2, June 2005.

[3] Bruce, K., Danyluk, A. and Murtagh, T. *Event-driven programming is simple enough for CS1*. Proceedings of the 6th annual conference on Innovation and technology in Computer Science Educations, 2001.

[4] Eney, C. and Hoyer, C. *Making a Difference on $10 a Day: Creating a "Women in CSE" Seminar Linked to CS1*. Proceedings of the American Society for Engineering Education, 2005.

[5] Horstmann, C. *Computing Concepts with Java Essentials* (third edition). John Wiley, 2003.

[6] Knuth, D. Selected Papers on Computer Science. CSLI, 1996.

[7] Koffman, E., Miller, P. and Wardle, C. *Recommended curriculum for CS1, 1984*. Communications of the ACM, October, 1984.

[8] Kölling, M. and Rosenberg, J. *Objects first with Java and BlueJ*. Proceedings of the thirty-first SIGCSE technical symposium on Computer Science Education, 2000.

[9] Lewis, J. and Loftus, W. Java Software Solutions (fourth edition). Addison Wesley, 2005.

[10] Pattis, R. *The "procedures early" approach in CS 1: a heresy*. Proceedings of the twenty-fourth SIGCSE technical symposium on Computer science education, 1993.

[11] Rasala, R., Raab, J. and Proulx, V. *Java power tools: model software for teaching object-oriented design*. Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, 2001.

[12] Reges, S. *Conservatively radical Java in CS1*. Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, 2000.

[13] Roberts, E. and Engel, G. (editors). Computing Curriculua 2001: Final Report of the Joint ACM/IEEE-CS Task Force on Computer Science Education. Los Alamitos, CA: IEEE Computer Society Press, December 2001. http://www.sigcse.org/cc2001/cs-introductory-courses.html.