BLOG@CACM

The *Communications* Web site, http://cacm.acm.org, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

twitter
Follow us on Twitter at http://twitter.com/blogCACM

DOI:10.1145/2093548.2093551 http://cacm.acm.org/blogs/blog-cacm

Knowledgeable Beginners

Bertrand Meyer presents data on the computer and programming knowledge of two groups of novice CS students.

Bertrand Meyer
"What 'Beginning' Students Already Know: The Evidence"
http://cacm.acm.org/blogs/blog-cacm/104215
January 25, 2011

For the past eight years I have been teaching the introductory programming course at ETH Zurich, using the "inverted curriculum" approach developed in a number of my earlier articles. The course has now produced a textbook, *Touch of Class*.[1] It is definitely "objects-first" and most importantly **contracts-first**, instilling a dose of systematic reasoning about programs right from the start.

Since the beginning I have been facing the problem that Mark Guzdial describes in his recent blog entry[2]: that an introductory programming course is not, for most of today's students, a *first* brush with programming. We actually have precise data, collected since the course began in its current form, to document Guzdial's assertion.

Research conducted with Michela Pedroni, who played a key role in the organization of the course until last year, and Manuel Oriol, who is now at the University of York, led to an ETH technical report: *What Do Beginning CS Majors Know?*[3] It provides detailed, empirical evidence on the prior programming background of entering computer science students.

BERTRAND MEYER
"An introductory programming course is not, for most of today's students, a first brush with programming."

Some qualifications: The information was provided by the students themselves in questionnaires at the beginning of the course; it covers the period 2003–2008; and it applies only to ETH. But we do not think these factors fundamentally affect the picture. We do not know of any incentive for students to bias their answers one way or the other; the 2009 and 2010 questionnaires did not show any significant deviation; and informal inquiries suggest that the ETH student profile is not special. In fact, quoting from the paper:

We did perform a similar test in a second institution in a different country [the U.K.]. The results from the student group at University of York are very similar to the results at ETH; in particular, they exhibit no significant differences concerning the computer literacy outcomes, prior programming knowledge, and the number of languages that an average student knows a little, well, and very well. A comparison is available in a separate report.[4]

Our paper[3] is short and I do not want to repeat its points here in any detail; please read it for a precise picture of what our students already know when they come in. Let me simply give the statistics on the answers to two basic questions: computer experience and programming language experience. We asked students how long they had been using a computer (see the first chart).

These are typically 19-year-old students. A clear conclusion is that we need not (as we naively did the first time) spend the first lecture or two telling them how to use a computer system.

The second chart summarizes the students' prior programming experience (again, the paper presents a more detailed picture).

Eighteen percent of the students (the proportion ranges from 13%–22% across the years of our study) have had no prior programming experience. Thirty percent have had programming experience, but not object-oriented; the range of languages and approaches, detailed in the paper, is broad, but particularly contains tools for programming Web sites, such as PHP. A little more than half have object-oriented programming experience; particularly remarkable are the 10% who say they have written an object-oriented system of more than 100 classes, a significant size for supposed beginners!

Guzdial's point was that *"There is no first"*; we are teaching students who already know how to program. Our figures confirm this view only in part. Some students have programmed before, but not all of them. Quoting again from our article:

At one end, a considerable fraction of students have no prior programming experience at all (between 13% and 22%) or only moderate knowledge of some of the cited languages (about 30%). At the present stage the evidence does not suggest a decrease in either of these phenomena.

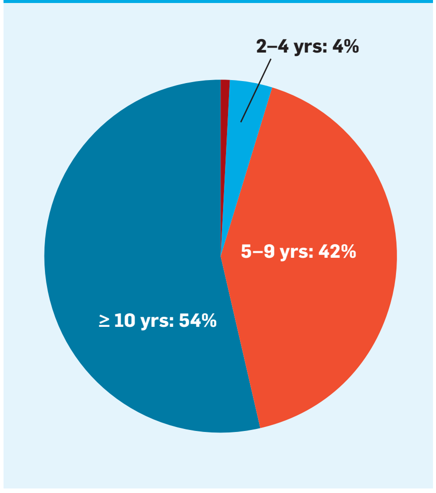
At the other end, the course faces a large portion of students with expertise in multiple programming languages (about 30% know more than three languages in depth). In fact, many have worked in a job where programming was a substantial part of their work (24% in 2003, 30% in 2004, 26% in 2005, 35% in 2006, 31% in 2007 and 2008).

This is our real challenge: How to teach an entering student body with such a variety of prior programming experience. It is difficult to imagine another scientific or engineering discipline where instructors face comparable diversity; a professor teaching Chemistry 101 can have a reasonable image of what the students know about the topic. Not so in computer science and programming. (In a recent discussion, Pamela Zave suggested that our experience may be comparable to that of instructors in a first-year language course, such as Spanish, where some of the students will be total beginners and others speak Spanish at home.)

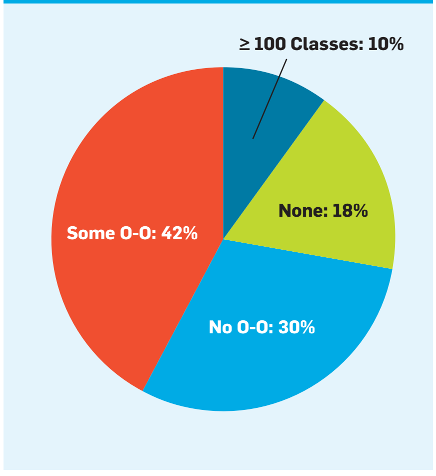
How do we bring something to all of them, the crack coder who has already written a compiler (yes, we have had this case) or an e-commerce site and the novice who has never seen an assignment before? It was the in-depth examination of these questions that led to the design of our course, based on the "outside-in" approach of using both components and progressively discovering their insides; it also led to the *Touch of Class* textbook[1], whose preface further discusses these issues.

A specific question that colleagues often ask when presented with statistics such as the above is why the experienced students bother to take a CS course at all. In fact, these students know exactly what they are doing. They realize their practical experience with programming lacks a theoretical foundation and have come to the university to learn the concepts that will enable them to progress to the next stage. This is why it is possible, in the end, for a single course to have something to offer to aspiring computer scientists of all bents—those who seem to have been furiously coding from kindergarten, the total novices, and those in between.

Computer experience.



Programming experience.



BERTRAND MEYER

"It is difficult to imagine another scientific or engineering discipline where instructors face comparable diversity; a professor teaching Chemistry 101 can have a reasonable image of what the students know about the topic. Not so in computer science and programming."

References

1. Bertrand Meyer, *Touch of Class: Learning to Program Well with Objects and Contracts*, Springer Verlag, 2009, see details and materials at http://touch.ethz.ch/.
2. Mark Guzdial, *We're too Late for 'First' in CS1*, http://cacm.acm.org/blogs/blog-cacm/102624.
3. Michela Pedroni, Manuel Oriol, and Bertrand Meyer, *What Do Beginning CS Majors Know?*, Technical Report 631, ETH Zurich, Chair of Software Engineering, 2009, http://se.ethz.ch/~meyer/publications/teaching/background.pdf.
4. Michela Pedroni and Manuel Oriol, *A Comparison of CS Student Backgrounds at Two Technical Universities*, Technical Report 613, ETH Zurich, 2009, ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/6xx/613.pdf.

Bertrand Meyer is a professor at ETH Zurich and ITMO (St. Petersburg) and chief architect of Eiffel Software.