The Prior Experience of Entering CS Students

Michela Pedroni
Chair of Software Engineering
ETH Zurich
8092 Zurich, Switzerland
pedronim@inf.ethz.ch

Manuel Oriol
Department of Computer Science
University of York
YO10 5DD, York, U.K.
manuel@cs.york.ac.uk

Bertrand Meyer
Chair of Software Engineering
ETH Zurich
8092 Zurich, Switzerland
meyer@inf.ethz.ch

ABSTRACT

One of the foremost issues for instructors of "Introduction to Programming" or "CS1" courses is the diversity of students' backgrounds – on one end of the range, a significant portion of students start their computing degree without prior programming expertise, while on the other end, many students have even worked in a job where programming was a substantial part. This diversity makes it difficult to adapt programming instruction to students' prior experience. The present article describes students' programming and computing experience when entering the ETH Computer Science bachelor program. It is based on the data of over 900 ETH students participating in the study in the past seven years and 77 students from University of York answering the questionnaire in 2008. The article reports on the analysis of changes over the years, presents a comparison between the data of ETH and York, and describes the pedagogical implications for courses and textbooks.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education

General Terms

Human Factors

Keywords

Student diversity, Programming experience, CS1

1. INTRODUCTION

It is difficult to think of another field than computer science in which, at the outset of the studies, instructors face heterogeneity in prior experience similar to what CS1 teachers handle – from students who have already worked as programmers in industry all the way to those who have no programming experience. The diversity of students' prior exposure to programming constitutes one of the challenges of teaching introductory programming.

This article presents a study that investigates entering CS students' prior experience in programming and computing. The study relies on a questionnaire that was issued in the Introduction to Programming course at ETH for the past seven years in the first weeks of class. The collected data make it possible to track changes over the years and to explore the stability of the results. To investigate whether the results are transferrable to other institutions, entering students of the University of York filled in the same questionnaire in 2008. A comparison with ETH data identifies similarities and differences between the two universities.

The following sections show related work, describe the setup of the study, and discuss the participant groups. The article also contains a discussion of the limitations of the study and presents comparisons of the questionnaire results for the groups dissected by topic (computer experience, programming experience, and programming languages). It proposes several measures to adapt to the student diversity and summarizes the results. Partial results were already reported in [13, 14, 15] and presented at the European Computer Science Summit 2009 [16].

2. RELATED WORK

A number of studies have provided information on students' prior computing and programming knowledge. They yield some important insights for the present work, but in most cases the issue of prior experience is subsidiary to the authors' main interest rather than focus of attention as in the present work. Sometimes the main issue is gender differences, as in [7, 12, 19] for CS majors and [11] for general students. In other cases the focus is on prediction of success in introductory programming courses as in [8, 10, 20, 17, 21]. None of these studies provide data to investigate differences between institutions or the stability of the situation concerning prior computing and programming knowledge of CS majors.

3. STUDY SETUP

Since Fall 2003, the first semester Computer Science majors at ETH attending the mandatory "Introduction to Programming" course have filled in a questionnaire in the first weeks of the semester. In the first iteration, the questionnaire was handed out on paper in class; in the following years it was available online.

The entering CS majors from University of York form the second group of participants. For these students, the data

have only been collected in 2008 with a voluntary online questionnaire.

The data from ETH make it possible to track changes over the years and help assess the stability of the situation concerning their prior programming and computing expertise. The data from York provide the means to compare the results of two institutions in two different countries and thus constitute a first step towards generalization of the multi-year results from ETH.

3.1 Questionnaire on student backgrounds

The questionnaire changed only slightly over time to include items that better capture students' answers. It contains three main sections: Computer literacy, prior programming experience, and knowledge of specific programming languages.

The items on computer literacy ask students whether they have a computer at home and similarly whether they have a laptop. Other items requested them to rate for how long they have been using the computer and for what they use it. Questions on their familiarity with the operating systems Windows, Linux, Mac OS, and BSD complement the computer literacy section of the questionnaire. These items use a four-point scale where students state whether they know the operating system *not at all*, *a little*, *well* or *very well*.

The questionnaire section on programming experience contains items on programming expertise (general and with object-oriented programming), where they have learned it, and the size of their largest projects (general and object-oriented) and development team.

The last part of the questionnaire collects data on the level of familiarity with various programming languages using a four-point scale (knows it *not at all*, *a little*, *well*, and *very well*).

3.2 Participants from ETH

At ETH, the students of seven iterations of the course Introduction to Programming answered the questionnaire. Out of the approximately 1340 students who took the course over the years, 914 handed back the questionnaire. The percentage of female students is between 5% and 16%.

Introduction to Programming is offered in the very first semester as the only computer science course and a required step for future computer science graduates on their way to a bachelor's and possibly a master's degree. The participants from ETH are CS majors (with a few exceptions) with a median age of 20, either new to the computer science study or repeating the first year because they have failed the final examination and retake Introduction to Programming as a preparation for their second try at the exams.

Most of the students who start a CS program at ETH come from one of the Swiss high schools where they graduated with the so-called "Maturity" degree. The Swiss high school system is decentralized: while a federal regulatory instrument sets general standards for the Maturity, each of the 26 cantons implements it with its own school laws. The Maturity system is selective – in 2009 less than 32% of all young permanent residents of Switzerland got a degree that allows them to study at a Swiss university [3]. In the computing area, most high schools offer introductory courses on computer applications (text processing, table calculations, web surfing), but very few teach computer science, or programming using a higher-level language. Until 2007, the Swiss high school regulations did not mention computer science; it recently became an optional supplementary subject, and implementation started in Fall 2008. It will be interesting to see how this affects the background of CS students.

3.3 Participants from University of York

All entering CS students at University of York (86 CS majors and 25 CS and Math majors) received the invitation to the questionnaire at the beginning of the semester. Out of the 101 students, 77 answered the call for participation.

As for ETH, most of the CS students at University of York enter directly from high school with a median age of 18. This is two years earlier than for ETH where the students go through a longer secondary education.

The British high school system has similar characteristics as the Swiss system concerning computing and computer science at high schools: there are no national regulations that require the schools to offer programming courses. The University of York has a selective admission process, but does not take prior computing and programming experience into account. Having taken a course on programming is therefore not a requirement for these students.

The percentage of female students at York is 9% – the same as for the ETH students in 2008.

4. THREATS TO VALIDITY AND LIMITATIONS

While we believe that many of the conclusions apply to the teaching of computer science anywhere, a number of specifics may limit generalization.

- The Swiss practice of selective high schools and the admission process at University of York, which screen the incoming students, may bias the sample of surveyed students towards higher competence.
- The absence of computer science in Swiss and British high schools (as opposed to many other countries) may bias the results in the reverse direction.
- Another threat to validity of the survey is that it does not measure students' prior experience objectively, but through their own self-appraisal. We do not know if this introduces a bias, and if so in what direction.
- Finally, the switch from a paper questionnaire, filled out in class at ETH in 2003, to a voluntary online form caused a decrease in participation. This introduces the risk of self-selection, another possible limitation. This threat to validity also applies to the data coming from the York participants.

  To minimize the risk of having an unrepresentative sample of students as participants to the questionnaire, we ask ETH students to rate their prior programming expertise again in the official end-of-semester course evaluation questionnaire required and administered by the university and handed out on paper during class. The results of that second test essentially coincide with the initial results, with the exception of a punctual discrepancy (23% of novices from the university questionnaire vs. 13%) for one single year, 2007.

An obvious potential limitation of this work is that it is mainly based on results from one institution. Although we

cannot authoritatively claim generalization to other universities and countries, the results from the student group at University of York are very similar to the results at ETH and provide a first indication that there are institutions with similar characteristics.

5. STATISTICAL TESTS

The collected questionnaires come from eight populations – seven classes of students taking Introduction to Programming at ETH in the years of 2003 until 2009 and one class from York in 2008. Based on these data, we can identify differences between the York and the ETH population and we can analyze changes over the years at ETH.

For the comparison of the ETH students to those of York, we apply Mann-Whitney-U tests [6, pp. 540–551] to the data of 2008. The Mann-Whitney test is a non-parametric test that makes it possible to identify significant differences between two independent samples (equivalent to the parametric t-test for independent samples, but also applicable to ordinal variables). To report the test results we use the scheme $U = ..., z = ..., r = ..., p < 0.05$, if the difference is significant at the 0.05-level. This means that a difference between the data of York and of ETH students is a chance finding with a probability of less than 5%. For non-significant results the scheme is $U = ..., z = ..., ns$. U is the test statistic of the Mann-Whitney test, z gives the z-score, and r reports the effect size. The effect size is an "objective and (usually) standardized measure of the magnitude of an observed effect" [6, p. 785]. This study uses Pearson's correlation coefficient r as measure for the effect size. We consider $r = 0.10$ a small effect, $r = 0.30$ a medium effect, and $r = 0.50$ a large effect.

To identify significant changes over the years at ETH, the analysis uses the Kruskall-Wallis-H test [6, pp. 559–571]. This non-parametric test makes it possible to identify significant differences between multiple independent samples. To report the Kruskall-Wallis test, we give the test statistic H, its degrees of freedom df and its significance p, $H(df) = ..., p < 0.05$, or $H(df) = ..., ns$, if it is non-significant.

The Kruskall-Wallis test only determines whether there are differences between the groups, but it does not indicate which groups are responsible for a significant outcome. To identify more precisely, which populations exhibit differences, we run Mann-Whitney-U tests as post hoc procedures on each pair of data sets. Using seven groups, this results in 21 Mann-Whitney tests as post hoc tests. We apply the Bonferroni-Holm correction [18, pp. 337–339] to keep the Type I error rate (false positives) down at the 5% level. The post hoc tests use the scheme of the Mann-Whitney tests for reporting.

6. COMPUTER LITERACY

Without knowing how to use a computer it is extremely difficult even to consider learning how to program. This is usually the first concern of a CS1 educator. In our setup, Figure 1 shows that this concern is no longer justified.

The figure confirms that students entering a computer science study are computer-literate. In fact, the majority has used computers for ten years or more. The data of 2008 from ETH and York do not differ significantly in this respect, $U = 4670, z = -0.30, ns$. With a median age of around 20 for ETH and 18 for York students, the computer has been part of their life for at least half of it.

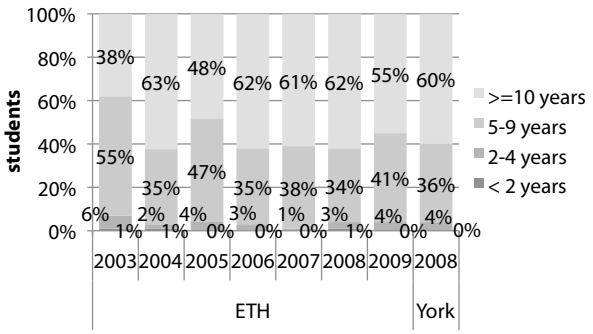


Figure 1: Time during which students have used computers

The application of the Kruskall-Wallis test indicates that there exists a significant difference between the seven classes of Introduction to Programming for the time spent with computers, $H(6) = 36.87, p < 0.05$. The post hoc tests show that this difference is mostly due to the 2003 population, which had significantly less exposure to the computer than did the students of the following years 2004-2009. This outcome repeats for most of the items of the questionnaire: if a significant difference exists between all the groups according to the Kruskall-Wallis test, then it is due to differences between the class of 2003 and the later classes. In the rest of the discussion, we will only report significant differences if they involve other years than 2003.

Table 1: Access to home computer and laptop

institution & year	computer	laptop
ETH		
2003	87%	56%
2004	98%	83%
2005	98%	81%
2006	96%	92%
2007	97%	75%
2008	95%	85%
2009	97%	87%
York		
2008	99%	90%

Consistently with the finding that the class of 2003 has less experience with the computer, the percentage of ETH students who have a desktop computer at home has risen from 87% in 2003 to percentages between 95% and 98% in the following years (see Table 1). Similarly, the percentages of students who own a laptop increased from 56% in 2003 to 75%-92% in the next years. The differences between the years are significant for desktop computers, $H(6) = 30.62, p < 0.05$, and for laptops, $H(6) = 84.06, p < 0.05$. The class of 2003, compared to the later classes, had significantly less access to desktop computers at home and a smaller portion of its students owned a laptop. Additionally, the number of students who own a laptop is significantly lower for the class of 2007 than for the students of the previous year, $U = 3501, z = -3.12, r = -0.23, p < 0.05$.

For York students, the situation is similar as for the ETH students in 2008: almost 99% of them have access to a computer at home and 90% own a laptop. These numbers indicate that students have similar access to technology at both institutions with respect to desktop computer, $U = 4566, z = -1.34, ns$, and laptops, $U = 4496, z = -1.02, ns$.

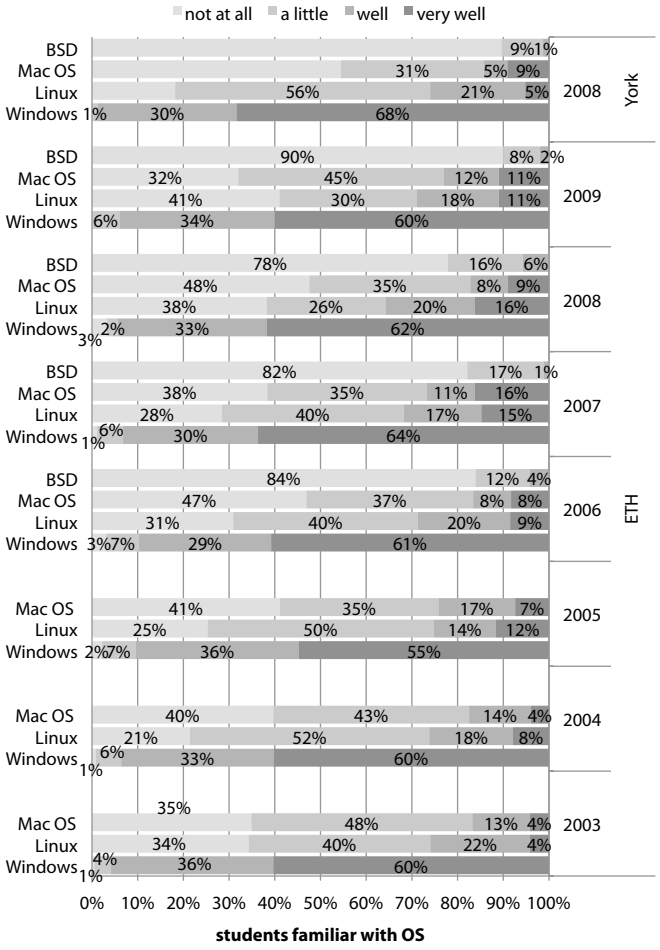


Figure 2: Students' familiarity with operating systems

Figure 2 shows students' familiarity with the Windows, Linux, Mac, and BSD operating systems. Note that the data for BSD have only been collected in the years 2005-2009. Windows is the operating system that most students know (around 97%-100%), followed by the Linux variants (59%-80%) and Mac OS (45%-68%). The percentage of students who state very high familiarity with Windows has been stable at around 60% over all the surveyed years, and there are no significant differences between the years for Windows, $H(6) = 2.17, ns$. The students of 2009 are less exposed to BSD than those of 2008, $U = 8371, z = -2.74, r = -.16, p < 0.05$, but the effect size is only small. The students at York also have significantly less experience with the operating system BSD than the 2008 ETH students, $U = 4131, z = -2.16, r = -0.15, p < 0.05$.

Concerning the specific tasks that students use the computer for, their main focus is on web-related tasks (see Table 2). All of them surf the web and write e-mails. Almost all of them also use the computer for text processing. About three quarters of all students play games and program, while about half of them design web pages, produce graphics, and do system administration and maintenance. There are no significant differences between the years 2005-2009 at ETH except for the item on web design and the item on programming. Significantly less students do web design in 2008 and 2009 than in the years 2005–2007. The students of 2009 also use their computer less for programming than those of 2007, $U = 5747, z = -3.16, r = -0.20, p < 0.05$.

ETH students use the computer significantly more for text processing, $U = 4409, z = -2.07, r = -0.15, p < 0.05$, and less for playing games, $U = 4055, z = -2.40, r = -0.17, p < 0.05$, than the York students. Note that the data for 2004 are missing.

Table 2: Technology tasks

task	ETH						York
	2003	2005	2006	2007	2008	2009	2008
surf the web	98%	100%	100%	99%	100%	100%	100%
write e-mails	98%	99%	100%	97%	99%	99%	100%
text processing	88%	96%	98%	94%	96%	99%	88%
play games	75%	74%	70%	73%	69%	73%	84%
graphics	44%	57%	68%	58%	53%	51%	46%
web design	47%	61%	68%	63%	48%	42%	42%
programming	65%	78%	79%	84%	77%	65%	68%
system admin.	43%	48%	53%	43%	50%	38%	39%

Generally, it seems that today's entering CS majors are computer-literate and use the computer regularly. The ETH and York students show only marginal differences with respect to their familiarity with the BSD operating system and their usage of the computer to play games and do text processing.

6.1 Interpretation

The significant differences between the class of 2003 and later classes indicate that the students of 2003 have been less exposed to computing and programming than later ones. This could be related to the increased spread of technology, which has reached the population of all ages. Today, most people use computers daily to read e-mails, surf the web and build up communities. Moreover, people attracted to computer science programs usually exhibit a strong interest in new media and technology.

Another explanation for the differences between 2003 and later years is a change in the Swiss high school regulations that shortened the duration of secondary education in certain cantons of Switzerland in the years of 2000-2003. This may have had an effect on the student body of 2003.

6.2 Teaching implications

The surveys do not indicate how deeply students understand the concepts behind computers and computer architecture. But the immediate lesson for CS1 instructors is that they do not need to fret about "computer literacy". Students are familiar with computers, and instructors can go straight into programming if this is the goal of the course.

7. PROGRAMMING EXPERIENCE

Table 3 shows the programming experience of students, broken down into the categories "no programming" (never programmed before), "no OO" (programmed, but never with an object-oriented language), "small project" (worked on object-oriented projects consisting of less than a hundred classes) and "large project" (worked on object-oriented projects with hundreds of classes – a sizable experience for supposed novices).

Table 3: Previous programming experience of students (*: $\geq$ 100 classes)

institution, year, gender	student numbers	no programming	some experience		
			no OO	some OO	
				small project	large project*
ETH					
2003					
total	222	22%	39%	34%	5%
male	203 (91%)	19%	39%	37%	5%
female	19 (9%)	53%	42%	5%	0%
2004					
total	127	14%	33%	43%	10%
male	117 (92%)	11%	34%	44%	11%
female	10 (8%)	50%	20%	30%	0%
2005					
total	95	18%	25%	42%	15%
male	81 (85%)	12%	26%	46%	16%
female	14 (15%)	50%	22%	21%	7%
2006					
total	97	19%	27%	43%	11%
male	84 (87%)	18%	25%	44%	13%
female	13 (13%)	23%	39%	38%	0%
2007					
total	88	13%	20%	59%	8%
male	84 (95%)	13%	19%	60%	8%
female	4 (5%)	0%	50%	50%	0%
2008					
total	124	18%	22%	43%	17%
male	113 (91%)	16%	22%	45%	17%
female	11 (9%)	46%	18%	18%	18%
2009					
total	161	18%	29%	48%	5%
male	136 (84%)	15%	27%	52%	6%
female	25 (16%)	32%	36%	32%	0%
York					
2008					
total	77	21%	27%	47%	5%
male	70 (91%)	19%	30%	46%	6%
female	7 (9%)	43%	0%	57%	0%

The number of female students participating in the questionnaire varied between 5% and 16%, reflecting the actual ratio of female first semester CS students. The table indicates that the figures do not differ markedly between the genders, except for the higher number of total beginners among females; however, this measure may not have any profound significance given the small size of the sample.

Figure 3 visualizes the results of Table 3 for students of both genders. It indicates that an increasing subset of the students starts with experience in object-oriented programming, while the percentage of those with no object-oriented programming experience has dropped. These changes over the years are only significant when taking into account the students of 2003, $H(6) = 24.56, p < 0.05$. The differences between the students of 2004 to 2009 are non-significant, $H(5) = 5.17, ns$.

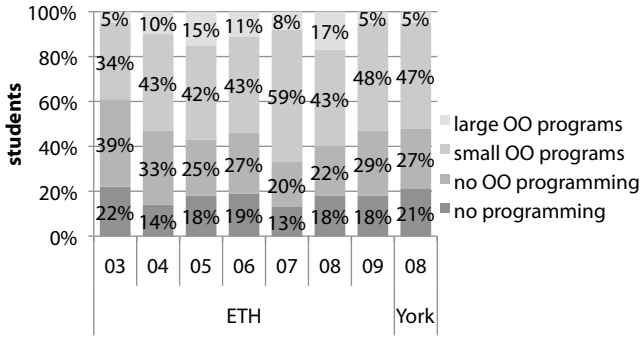


Figure 3: Programming experience

There is no identifiable change of trend over the years in the number of novices, which hovers around one-sixth to one-fifth of each class.

The distribution of prior programming expertise of the York students is consistent with the ETH data. There seems to exist a general scheme of around 20% novices, between 20-30% of non-OOP programmers, and around 40%-50% with experience on small object-oriented programs. For York, the number of students who have programmed large object-oriented applications is smaller than for ETH, but the Mann-Whitney test shows no significant differences for the two groups, $U = 4159, z = -1.63, ns$.

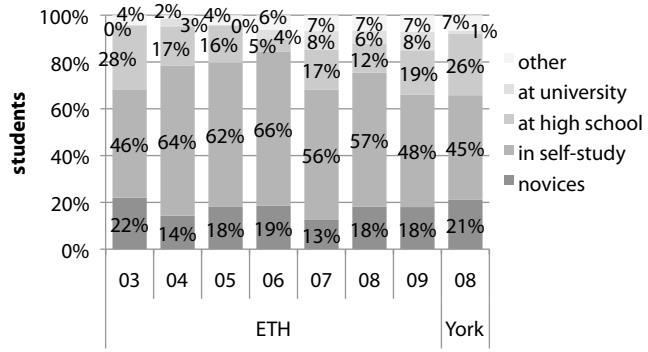


Figure 4: Where students learned to program

Another item on the questionnaires asked students where they have learned to program (see Figure 4). On average over the seven years, 55% of the ETH students stated that they learned programming by themselves; 18% are novices; only 18% took a programming course at high school and the remaining 9% learned it at university, at work or on another occasion (such as courses at an evening school).

Similarly to the ETH students, most students at York learned programming in self-study (45%); 26% took a programming course at high school, and 8% learned it at an-

other occasion. Compared to the ETH students, 8% more students have studied programming at high school, but, as for the ETH population, the most frequent case is having it learned in self-study. Statistical tests did not find a significant difference between the two populations, $U = 2755, z = -1.51, ns$.

7.1 Interpretation

A possible reason for the lower exposure to programming of the class surveyed in 2003 could be that this was still just "after the Internet bubble burst", after which more students have been attracted to computer science by genuine interest. Observations that would seem to support such a hypothesis include: the highest percentage of novice programmers (22% including both genders) for the year of 2003; the above-average numbers of CS enrollments in that year (although an alternative explanation for that particular phenomenon could be a change that occurred in the Swiss high school system); and our own informal observation that students in subsequent years seemed more genuinely interested in CS.

The growth in object-oriented language experience is probably due to the increasing spread of object-oriented languages such as Java (see also Section 8).

Many students have prior programming experience when entering university. Interestingly, most of them learned it in self-study; only one fourth to one sixth of them have taken a programming course at high school. Although partly influenced by the absence of mandatory programming courses at high schools, this result shows how broadly part of computer science has reached some of the world at large, particularly the younger segments of the population.

7.2 Teaching implications

The evidence on prior object-oriented language use indicates that today object-orientation is a given and needs no particular apology or justification. Tempering this lesson coming from the questionnaire data is a more subjective observation from our informal interactions with students: many do not fully grasp the more sophisticated properties of object technology, such as polymorphism, dynamic binding and other architectural techniques. It seems more useful to explain these concepts in depth than to take pains to justify the use of objects.

Another important conclusion arises from studying the other end of the data: the persistence of the "no prior programming" 15%-20% minority. It raises significant challenges for teachers, especially when assessed against the only slightly lower percentage of those who have programmed fairly large object-oriented systems. The variety of prior programming expertise is, in our experience, one of the largest obstacles facing introductory programming teaching today.

8. PROGRAMMING LANGUAGES

As part of the questionnaire, students rated 15 programming languages (ranging from Java, PHP and C++ to Fortran, Eiffel and Python; for a full list see Figure 8) whether they know it *not at all*, *a little*, *well* or *very well*. The answers to these questions (Table 4) reveal that a typical student of any of the two institutions knows – in his or her self-evaluation – two to three of the languages a little and at least one of the languages well.

Table 4: Average (and median) number of languages known

institution & year	a little	well	very well
ETH			
2003	1.8 (2)	1.0 (1)	0.2 (0)
2004	3.2 (2)	1.1 (1)	0.6 (0)
2005	3.2 (3)	1.4 (1)	0.6 (0)
2006	2.8 (3)	1.2 (1)	0.7 (0)
2007	2.6 (2)	1.3 (1)	0.6 (0)
2008	2.4 (2)	1.2 (1)	0.5 (0)
2009	2.9 (3)	1.1 (1)	0.5 (0)
York			
2008	2.0 (2)	1.0 (1)	0.5 (0)

As to the number of programming languages students know well or very well, Figure 5 confirms that almost half of the current students have sound proficiency in two or more languages and that at least one third of all students have not really mastered any of the languages (the numbers include the students who stated being novice programmers). These figures did not change significantly for ETH students in 2004 to 2009, $H(5) = 4.9, ns$. There are more students who claim to know only one language well or very well at York than at ETH; at ETH, the percentage of students who claim to know three or more languages well or very well is higher than at York. These differences, however, are not statistically significant, $U = .92, z = -.10, ns$.

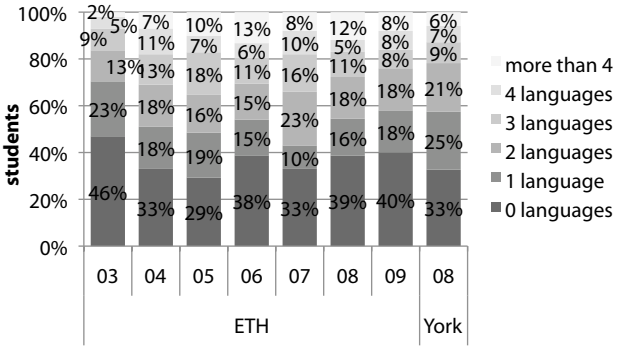


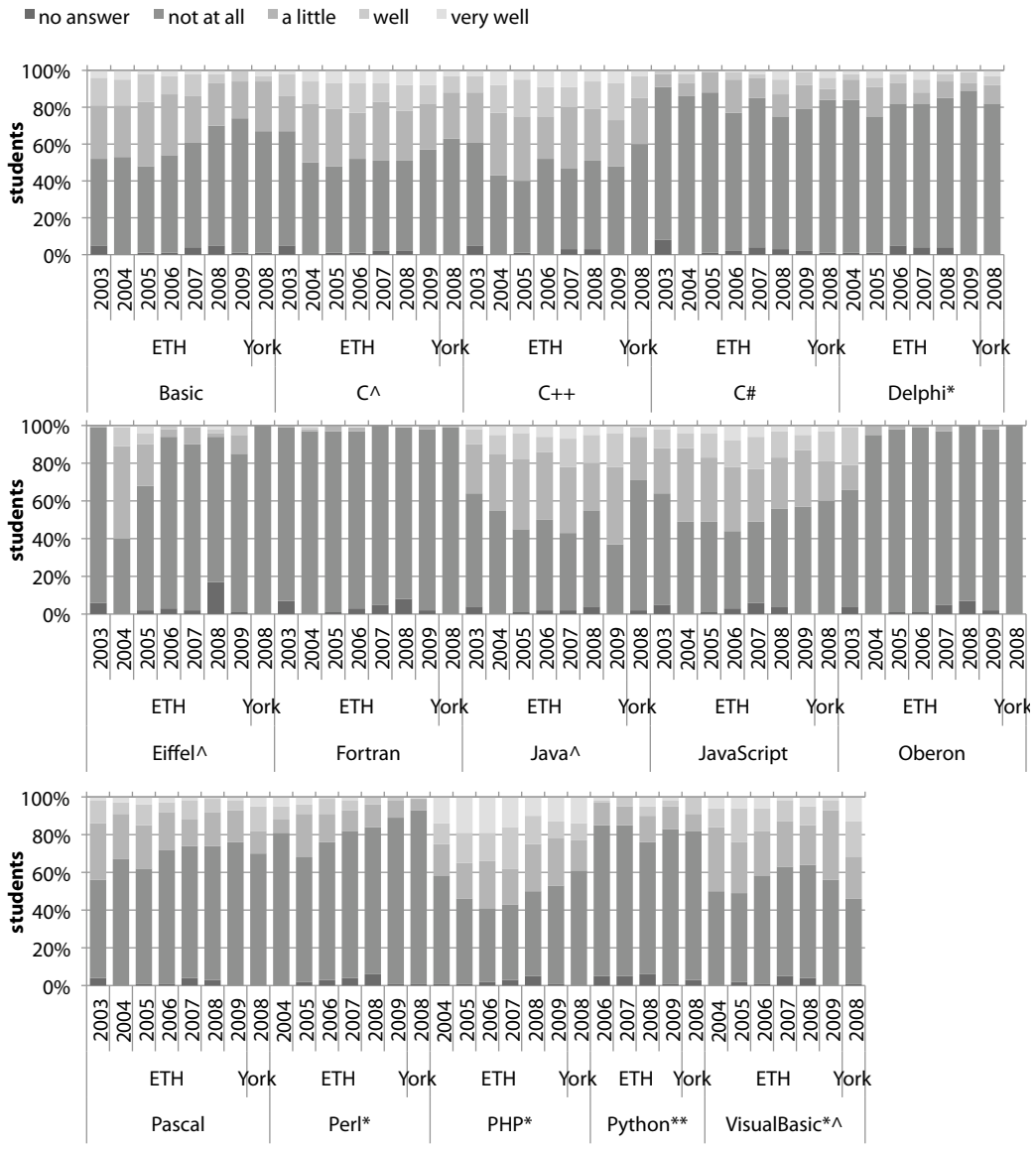
Figure 5: Number of languages known well or very well

Questionnaire items on the level of familiarity of the 15 programming languages help answer additional questions: (1) What are the most widely known languages among them? (2) Are there languages with growing or dropping popularity with this particular population?

Figure 8 shows the 15 programming languages and the percentages of students with the four levels of familiarity (knowing the language in question *not at all*, *a little*, *well* and *very well*). Some of the languages, marked * and **, were only included in the survey after the first iterations. The analysis takes into consideration the answers from all students (including programming novices).

Taken over all years for ETH students, Java and the web scripting language PHP are the most popular with high numbers of students who state they know it very well and only few students who do not know it at all. Other popular languages are C/C++, JavaScript, and Basic/VisualBasic.

The top three languages (i.e. the languages where the



* marks missing data of 2003
** marks missing data of 2003-2005
ˆ marks significant differences between York and ETH students

Figure 6: Familiarity of students with programming languages

least ETH students state that they do not know it at all), separated by year, include most of the languages rated as most known totaled over the years (see Table 5 and Figure 8). C++ is an evergreen – it appears almost every year in the list of the three top languages. Since 2005, Java, JavaScript and PHP also strengthened their position and for the last three years around 50% of all students have worked with PHP, JavaScript, and/or Java before starting to study CS.

VisualBasic is the top language at York followed by the scripting languages JavaScript and PHP (see Table 5). Indeed, while ETH and York students do not exhibit significant differences in their general programming backgrounds, they differ significantly in some of the specific programming languages that they worked with before entering university.

The programming languages in Figure 8 marked with ˆ differ significantly for the two populations. Examples of such languages include VisualBasic, $U = 3590, z = -2.84, r = -0.20, p < 0.05$, and Java, $U = 3610, z = -2.61, r = -0.19$, $p < 0.05$.

The data of ETH collected over multiple years make it possible to analyze changes in the popularity of programming languages. The most popular programming languages such as PHP and Java belong to the list of programming languages with increasing popularity amongst the students. Figure 7 shows the most popular programming languages

Table 5: Most popular programming languages

institution & year	1st place	2nd place	3rd place
ETH			
2003	Basic	Pascal	C++
2004	Eiffel	C++	JavaScript
2005	C++	Java	PHP
2006	PHP	JavaScript	Java
2007	PHP	Java	JavaScript/C++
2008	PHP	C++	C
2009	Java	C++	PHP
York			
2008	VisualBasic	JavaScript	PHP

that have a rising tendency in the percentage of ETH students who state to know it *a little*, *well*, or *very well*, i.e. the percentage of ETH students who do not know the programming language at all has decreased since 2003. The programming languages Basic, Pascal, and VisualBasic exhibit a decreasing trend in ETH students' level of familiarity. However, most of these results are not significant. The Kruskall-Wallis test applied to the ETH populations of 2003-2009 exhibits only three languages with significant differences involving other years than 2003. In 2008 and 2009, significantly less students knew Basic than in some of the years between 2004 and 2006, $H(6) = 38.83, p < 0.05$. The students of 2004 and 2005 generally state to have significantly more experience with Eiffel than the students of most other years, $H(6) = 215.98, p < 0.05$. Additionally, in 2009 less students had experience with Perl than in 2005, $U = 5902, z = -4.137, r = -0.26, p < 0.05$.

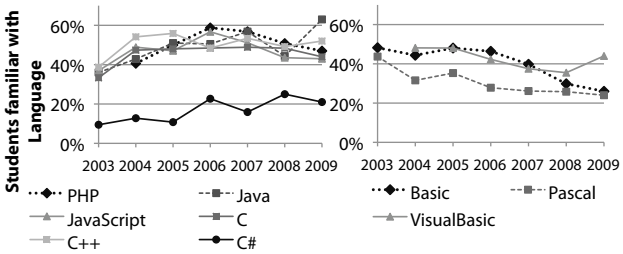


Figure 7: Evolution of popularity of programming languages

8.1 Interpretation

The popularity of languages such as JavaScript and PHP most likely reflects that many students' prior experience has been with web applications. This fits into the observation that today's students are very web-oriented using the computer for writing e-mails and surfing on the web.

Student backgrounds of the two populations at York and ETH only differ with respect to the specific programming languages that are best known (such as Java and VisualBasic). This could indicate that the popularity of programming languages is location-specific and possibly due to the varying spread of the programming languages at the two countries' high schools. But at this point, the collected data are too sparse to verify this conjecture.

Note that our results are limited to the 15 languages itemized in the questionnaire: a student may know additional languages.

8.2 Teaching implications

These results show that, when teaching introductory programming, we need to take into account that the number of students who need to learn programming almost from scratch is higher than the 10% to 20% who have never programmed before.

In particular, it may well be that students whose programming has mostly been with Web applications in PHP or JavaScript are adept at writing user interface operations, but only have superficial experience with loops, recursion, data structures and other standard computer science techniques. While the questionnaire does not test this conjecture, it is definitely supported by informal observations. If it is correct, we should not consider that proficiency at GUI and Web programming implies proficiency at concepts and skills of professional software development, meaning that we need to take extra care with the teaching of fundamental topics.

9. EFFECT ON TEACHING

The analysis of the questionnaire on student backgrounds illustrates the full spectrum of student backgrounds that CS1 instructors face. This section presents measures proposed to adapt to such a student body.

Adapting the course material. As a first and simple option, if we want students with prior knowledge to understand courses better, we must connect to that knowledge. This can help adapt the course to students' needs; when introducing a concept, for example, instructors can provide references to its counterpart in the most known programming languages. They may consider going further and organizing special exercise groups for students with in-depth experience with a specific programming language.

Adapting the teaching methodology. Because the majority of CS1 students already know a programming language, it seems more natural to offer access to the whole libraries and to a complex development environment, thus letting the more curious students explore a richer environment. This is the technique used in the Inverted Curriculum approach [13]. While more novice students content themselves with the library's APIs, their advanced colleagues may explore the library's internals, discover the more advanced aspects, and enhance their competence through imitation and inspiration.

Another possible measure to tackle differences in prior programming experience is the use of a programming language that only few students have previously encountered in CS1. Interestingly, both institutions use such a language to "level students out" and not favor one student over another. While this is not the full solution to the problem of student diversity, it seems to be a reasonable way of ensuring that all students have a fair treatment.

Offering extra lessons. Students who had learned a programming language prior to the CS1 course are, overall, more successful than novices [10, 17]. To confirm that these findings apply to our setting, we use the ETH data of two

years (2004 and 2008), for which we are able to associate grades in the final exam to the questionnaire data. Calculating Spearman's correlation coefficient reveals that the grades of the students are significantly related to their previous programming knowledge (grouped into the categories "no programming", "no OO", "small OO programs", and "large OO programs"), $r_s = .31, p < 0.001$. The average and median grades dissected by groups reflect this relationship. Note that the Swiss grades range from 1 (lowest grade) to 6 (highest grade) in 0.25 increments; 4 is the lowest passing grade.

Table 6: Grades by prior programming knowledge

	Mean	Median
No programming	4.18	4.25
No OO	4.49	4.75
Small OO programs	4.80	5.00
Large OO programs	5.21	5.25

It is likely that the extra experience with other programming languages provides intellectual preparation for mastering the intricacies of software development, for which novices enjoy no counterpart. To redress this imbalance, it may be interesting to allow novice students to take extra lessons on programming either before the semester starts (such as in a CS0 course [2]) or during the semester.

Making student groups. The differences in students' prior programming expertise justify the discussion of offering two or more courses targeted to the various competence levels. This would require the development of competence models and associated programming language independent tests, which guarantee objective assessment of prior knowledge. The competence model for object interaction developed by Bennedsen and Schulte [1] could serve as a starting point. It describes a taxonomy consisting of four levels. The levels range from understanding simple object interactions (such as feature calls and object creation) between a few objects on the lowest level to understanding the effects of inheritance and dynamic binding on dynamic polymorphic object structures on the highest level. The course for advanced programmers could then build upon existing previous knowledge and cover ancillary material while the course for novices could take a slower approach in introducing new concepts. One of the main issues with such a solution is probably scarceness of resources and increased costs. A reduced solution could split up the course partially (for example only for the lab or tutoring sessions such that certain parts can be adapted to previous knowledge).

Individualized instruction. Going one step further, individualized instruction seems well suited to handle the student diversity. One of the main characteristics of individualized instruction is that students proceed at their own pace, so that the differences in prior knowledge can be handled. Individualized instruction is related to other teaching methods such as programmed instruction, mastery learning, self-controlled learning, and computer-assisted instruction. Several variants of individualized instruction find applications in teaching introductory programming [4, 5, 9].

10. CONCLUSIONS

We will continue to track students' prior experience, which we view as an indispensable tool for tuning courses to the real students of the 21st century. The questionnaires have proved extremely useful in this endeavor.

The analysis of the data collected over seven years found that the student body of 2003 significantly differs from the later classes. If the data set of 2003 is ignored, then the situation appears stable with the exception of single pointed changes in the percentage of students owning a laptop, the portion of students using the computer for web design and programming, their familiarity with the operating system BSD, and their knowledge of Basic and Perl. The only largely unstable item is on the knowledge of Eiffel. This confirms that the ETH introductory programming course has been and – given the mostly stable situation – will be faced with a very diverse student body.

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 (around 30%). The evidence so far 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 (24% in 2003, 30% in 2004, 26% in 2005, 35% in 2006, 31% in 2007 and 2008, and 25% in 2009).

An increasing percentage of students who have programming experience used an object-oriented language; correspondingly, fewer students take the course without prior exposure to object-oriented programming.

If we try to picture the typical entering CS student at any of the two institutions, he (being typical, the student is most likely a "he") is between 18 and 20 years old and knows one programming language in depth and another two to three languages slightly. He has a computer with the Windows operating system at home and most likely a laptop. He has a long experience with computers and uses it mostly for web surfing, writing e-mails, and text processing. He has learned programming by himself and uses VisualBasic, Java, C++, C, or a web-related programming language such as PHP or JavaScript.

The data presented in this article show that the entering CS students at ETH and those at University of York start with very similar backgrounds, in particular concerning prior programming experience, and the number of programming languages that they know. They only differ in the specific programming languages that they know and single items concerning computer literacy.

There are several possible factors for this similarity. First, courses on computer science are optional in the high school systems of both countries. In addition, ETH and University of York are both competitive and recruit nationwide. Third, neither ETH nor University of York requires computer science or programming expertise for admission. While the University of York has a selective admission process based on other criteria such as Math grades, ETH accepts anybody with a Swiss Maturity degree. The selection at ETH happens through the selective Swiss high schools, ETH's reputation as a top university, and the first year exams.

The outcome suggests that generalization of the results is possible, but may be limited to other universities of countries

with similar regulations for admission and for high school courses on computer science. To address the issue of generalization fully, it is necessary to broaden the scope of the study to more universities and countries.

11. REFERENCES

[1] J. Bennedsen and C. Schulte. A competence model for object-interaction in introductory programming. In *18th Workshop of the Psychology of Programming Interest Group, PPIG*, University of Sussex, September 2006.

[2] C. Dierbach, B. Taylor, H. Zhou, and I. Zimand. Experiences with a CS0 course targeted for CS1 success. *SIGCSE Bulletin*, 37 (1):317 – 320, 2005.

[3] Eidgenössisches Bundesamt für Statistik, Schweiz. Maturitätsquote nach Maturitätstyp und Kanton. Online at http://www.bfs.admin.ch, June 2010.

[4] H. H. Emurian. Teaching Java^{TM}: Managing instructional tactics to optimize student learning. *International Journal of Information & Communication Technology Education*, 3(4):34 –49, 2007.

[5] L. Faessler, H. Hinterberger, M. Dahinden, and M. Wyss. Evaluating student motivation in constructivistic, problem-based introductory computer science courses. In *E-Learn 2006: Proceedings of the World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education*, Honolulu, Hawaii, USA, October 2006.

[6] A. Field. *Discovering Statistics Using SPSS (Introducing Statistical Methods series)*. Sage Publications Ltd, 3. edition, 2009.

[7] A. Fisher, J. Margolis, and F. Miller. Undergraduate women in computer science: experience, motivation and culture. *SIGCSE Bull.*, 29(1):106–110, 1997.

[8] A. Gomes and A. Mendes. A study on student's characteristics and programming learning. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008*, pages 2895–2904, Vienna, Austria, June 2008. AACE.

[9] T. Grandon Gill and C. F. Holton. A self-paced introductory programming course. *Journal of Information Technology Education*, 5:95 – 105, 2006.

[10] D. Hagan and S. Markham. Does it help to have some programming experience before beginning a computing degree program? *SIGCSE Bull.*, 32(3):25–28, 2000.

[11] M. E. Hoffman and D. R. Vance. Computer literacy: what students know and from whom they learned it. *SIGCSE Bull.*, 37(1):356–360, 2005.

[12] E. M. Madigan, M. Goodfellow, and J. A. Stone. Gender, perceptions, and reality: technological literacy among first-year students. *SIGCSE Bull.*, 39(1):410–414, 2007.

[13] M. Pedroni and B. Meyer. The inverted curriculum in practice. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 481–485, New York, NY, USA, 2006. ACM Press.

[14] M. Pedroni, B. Meyer, and M. Oriol. What do beginning CS majors know? Technical Report 631, ETH Zurich, July 2009.

[15] M. Pedroni and M. Oriol. A comparison of CS student backgrounds at two universities. Technical Report 613, ETH Zurich, July 2009.

[16] M. Pedroni, M. Oriol, and B. Meyer. Student diversity in CS1. Presented at European Computer Science Summit – ECSS 2009, 5th Annual Informatics-Europe Meeting, Paris, October 2009.

[17] J. S. Rosenschein, T. Vilner, and E. Zur. Work in progress: programming knowledge – does it affect success in the course "introduction to computer science using Java". *Frontiers in Education, 2004. FIE 2004. 34th Annual*, 1:T2H/3–T2H/4, Oct. 2004.

[18] B. Rüger. *Test- und Schätztheorie – 2. Statistische Tests*. Oldenbourg, 2002.

[19] M. G. Sackrowitz and A. P. Parelius. An unlevel playing field: women in the introductory computer science courses. *SIGCSE Bull.*, 28(1):37–41, 1996.

[20] P. Ventura and B. Ramamurthy. Wanted: CS1 students. No experience required. *SIGCSE Bull.*, 36(1):240–244, 2004.

[21] B. C. Wilson. A study of factors promoting success in computer science including gender differences. *Computer Science Education*, 12(1-2):141–164, 2002.