A Comparison of CS Student Backgrounds at Two Universities

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

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

ABSTRACT
It is difficult to think of another field that, at the outset of the studies, has to face heterogeneity in prior experience similar to what CS1 instructors handle. This student diversity in Computer Science seems to be a global phenomenon. The present work investigates to what extent the student backgrounds concerning prior programming and computing experience are similar both at ETH Zurich, Switzerland and at University of York, United Kingdom. The results are surprising: there exist no significant differences between the entering CS majors at the two institutions with respect to their prior programming expertise, computing literacy and number of languages that they know. The only differences that proved significant were in the specific programming languages with which students had worked before entering the university.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education – computer science education.

General Terms
Human Factors.

Keywords
programming experience, student diversity.

1. INTRODUCTION
Computer Science in Switzerland and in the United Kingdom has a special status among the major topics that students may pick at university: the courses at high school that prepare students for a Computer Science study and provide them with prior knowledge are not mandatory and at some high schools the subject is not even taught. Both at ETH Zurich and at University of York there is no prior experience on programming required to register, although in fact many entering CS majors have programmed before. The result is a very diverse body of students. This observation is neither new nor institution-specific as indicated through [6] and captures a universal challenge that CS1 instructors face. The question is then whether the composition of the student body can be characterized more precisely and whether these characteristics transfer to other institutions and countries.

This article takes a look at students’ prior experience in programming and the issue of transfer by comparing results of a study run both at ETH Zurich and at University of York. The outcome of the study shows that the students at the two institutions indeed have similar prior knowledge: approximately 20% of the students have no prior experience in programming, 30% have moderate knowledge, 20% have good knowledge of at least one programming language, and 30% claim that they know three or more programming languages in depth.

This exhibits the whole spectrum of knowledge. It impacts heavily on the teaching that students accept. Interestingly, to teach introductory programming both institutions use languages with which students have very little experience to “level students out” and not advantage 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.

Section 2 shows related work. Section 3 details the setup of the study and discusses the participant groups and limitations of the study. Sections 4, 5, and 6 present comparisons of the questionnaire results for the groups dissected by topic (computer experience, programming experience, and programming languages). Section 7 summarizes the results. Section 8 presents conclusions and future work.

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.

Hoffmann and Vance [5] give an overview of the “technology tasks” that university students are able to handle and whom they learnt it from. Madigan et. al. [8] describe computer literacy with a focus on gender differences.

Most studies report on prior programming knowledge as a possible predictor of success in a CS1 course. Hagan and Markham [4] report that in 2000 C/C++ and Pascal/Delphi were the most prominent programming languages for their students to have previously encountered as instructional languages. In their study they confirm that prior programming experience at least initially is beneficial for the students of their course. They also find that the number of programming languages that the students previously encountered is systematically related to this outcome, but not the specific languages themselves. In 2002, Wilson [14] showed that having attended a formal class in programming has a positive correlation with success in their course. The study of Rosenschtein et. al. [11] in 2003 shows that in their course Pascal is the programming language that most of their students are familiar with. In their setting knowledge of Pascal, C, and
Assembler is a predictor for success, but not knowledge of other languages such as Prolog, C++ or Java. The studies mentioned above generally confirm prior programming knowledge as a predictor of success while the study by Ventura and Ramamurthy [10] in 2004 comes to the conclusion that for their course (design-centered, objects-first, graphical) prior programming knowledge is no predictor of success. Gomes and Mendes [3] in 2008, with participants from two Informatics Engineering courses, show the familiarity of students with the programming languages C, Pascal, Java, and Python. From this data, they find that previous programming knowledge is a predictor of success in their (objects-first) course.

Prior programming knowledge has also been studied to gain insights into gender differences. Sakrowitz and Parelius [12] show in their study of 1994 that women are less likely to have taken a programming course before starting to study CS and that their familiarity with Pascal, Basic and C is lower than for their male colleagues. Fisher et. al. [2] confirm these findings.

Lahtinen et. al. [7] use the data on prior programming knowledge of 559 students and 34 teachers in connection with identifying difficult concepts when learning to program. Their study shows that almost 60% of the students attest themselves of having prior programming experience.

None of these studies provide data to investigate differences between institutions on the basis of prior computing and programming knowledge of CS majors.

3. STUDY SETUP

Two groups of participants answered the questionnaire.

The students of the Introduction to Programming course at ETH have, since fall 2003, filled in a questionnaire in the first weeks of the semester. This data makes it possible to track changes over the years and helps assess the stability of the situation concerning their prior programming and computing expertise. Partial results of this questionnaire were already presented in [5].

The entering CS majors from University of York form the second group of participants. For these students, the data has only been collected in 2008. The comparison between ETH and York students builds a first step towards generalization of the multi-year results from ETH.

3.1 Participants from ETH

The participants from ETH generally are CS majors (with a few exceptions), either new to the computer science study or repeating the first year because they have failed the final examination. These students typically decide to retake Introduction to Programming as a preparation for their second try at the exams.

“Introduction to Programming” is offered in the very first semester amongst other courses on logic, linear algebra, analysis, and statistics. It is a required subject for entering CS majors on their way to a bachelor’s and possibly a master’s degree.

Most of the students that start a CS program at ETH come from one of the Swiss high schools. The local high school system is decentralized: while a federal regulatory instrument sets general standards for the high school degree, each of the 26 cantons implements it with its own school laws. In the computing area, most high schools offer introductory courses on computer applications (text processing, table calculations, web surfing), but very few teach programming using a higher-level language.

The number of students that participated in the courses over the six years is approximately 1130 (250 in 2003, 180 in 2004, 170 in 2005, 160 in 2006, 170 in 2007, and 200 in 2008). The percentage of female students is between 5% and 15%.

In the first iteration, the questionnaire was handed out on paper in class; in the following years it was available online.

3.2 Participants from University of York

The invitation to the questionnaire was sent out at the beginning of the semester to all entering CS students at University of York (86 CS majors and 25 CS and Math majors). Out of the 101 available 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. Compared to ETH this is two years earlier and most likely due to longer secondary education in Switzerland, where most of the students are 19 when they graduate from high school.

The British high school system has similar characteristics concerning computing and computer science at high schools: there are no 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 the same as for the ETH students in 2008, 9%.

3.3 Threats to validity

One 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.

The questionnaire in a voluntary online form introduces the risk of self-selection, another possible limitation.

4. COMPUTER LITERACY

![Figure 1: Time during which students have used computers](image)

The data visualized in Figure 1 confirms that students entering a computer science study are computer literate. In fact, the majority has used computers for ten years or more. With a median age of
around 20 for ETH and 19 for York students, the computer has been part of their life for at least half of it.

In line with these findings, the percentage of students that have a desktop computer at home is between 95% and 98% in the years of 2004-2008. For York students, the situation is similar: almost 99% of them have access to a computer at home. 90% of the students at York own a laptop while at ETH these percentages increased from 56% in 2003 to 75%-93% in the next years. These numbers indicate that students have similar access to technology at both institutions. Indeed, statistical tests show no significant difference between the two groups concerning their computer literacy.

5. PROGRAMMING EXPERIENCE

Table 1 shows the programming experience of students, broken down into the categories “no programming” (never programmed before), “no O-O” (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 O-O projects with hundreds of classes — a sizable experience for supposed novices).

<table>
<thead>
<tr>
<th>Year</th>
<th>Gender</th>
<th>Number of Students</th>
<th>No Programming</th>
<th>No O-O</th>
<th>Small Project</th>
<th>Large Project (&gt;100 classes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>Total</td>
<td>222</td>
<td>22%</td>
<td>22%</td>
<td>19%</td>
<td>19%</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>203 (91%)</td>
<td></td>
<td>39%</td>
<td>34%</td>
<td>5%</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>19 (9%)</td>
<td></td>
<td>39%</td>
<td>5%</td>
<td>5%</td>
</tr>
<tr>
<td>2004</td>
<td>Total</td>
<td>127</td>
<td>14%</td>
<td>33%</td>
<td>11%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>117 (92%)</td>
<td></td>
<td>34%</td>
<td>26%</td>
<td>11%</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>10 (8%)</td>
<td></td>
<td>20%</td>
<td>30%</td>
<td>0%</td>
</tr>
<tr>
<td>2005</td>
<td>Total</td>
<td>95</td>
<td>18%</td>
<td>25%</td>
<td>12%</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>81 (85%)</td>
<td></td>
<td>26%</td>
<td>22%</td>
<td>7%</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>14 (15%)</td>
<td></td>
<td>22%</td>
<td>21%</td>
<td>7%</td>
</tr>
<tr>
<td>2006</td>
<td>Total</td>
<td>97</td>
<td>19%</td>
<td>27%</td>
<td>18%</td>
<td>13%</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>84 (87%)</td>
<td></td>
<td>25%</td>
<td>23%</td>
<td>13%</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>13 (13%)</td>
<td></td>
<td>39%</td>
<td>38%</td>
<td>0%</td>
</tr>
<tr>
<td>2007</td>
<td>Total</td>
<td>88</td>
<td>13%</td>
<td>20%</td>
<td>13%</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>84 (95%)</td>
<td></td>
<td>19%</td>
<td>59%</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>4 (5%)</td>
<td></td>
<td>50%</td>
<td>50%</td>
<td>0%</td>
</tr>
<tr>
<td>2008</td>
<td>Total</td>
<td>124</td>
<td>18%</td>
<td>22%</td>
<td>16%</td>
<td>17%</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>113 (91%)</td>
<td></td>
<td>22%</td>
<td>46%</td>
<td>18%</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>11 (9%)</td>
<td></td>
<td>18%</td>
<td>18%</td>
<td>18%</td>
</tr>
</tbody>
</table>

The numbers from York are very similar to the numbers from ETH (see Table 1). There seems to be a general scheme containing 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 that have programmed large OO applications is smaller than for ETH, but the Mann-Whitney tests shows no significant differences for the two groups.

Given the small numbers of female students at both institutions, it is not possible to draw conclusions from their data.

Figure 2 visualizes the results of Table 1 for students of both genders. It indicates that the distribution of prior programming expertise of York students indeed fits into the picture of the data of ETH students.

6. PROGRAMMING LANGUAGES

As part of the questionnaire, students were asked to rate 15 programming languages (ranging from Java, PHP and C++ to Fortran, Eiffel and Python; for a full list see Figure 5) whether they know it not at all, a little, well or very well. The answers to these questions (Table 2) reveal that an average student knows — in his or her self-evaluation — two to three of the languages a little and at least one of the languages well. This is the same for the York participants.
well of familiarity (knowing the language in question percentages of students

Figure 5: Overall popularity of programming languages

Table 2: Average (and median in parenthesis) number of languages known

<table>
<thead>
<tr>
<th></th>
<th>year</th>
<th>a little</th>
<th>well</th>
<th>very well</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH</td>
<td>2003</td>
<td>1.8 (2)</td>
<td>1.0 (0)</td>
<td>0.2 (0)</td>
</tr>
<tr>
<td></td>
<td>2004</td>
<td>3.2 (2)</td>
<td>1.1 (0)</td>
<td>0.6 (0)</td>
</tr>
<tr>
<td></td>
<td>2005</td>
<td>3.2 (3)</td>
<td>1.4 (1)</td>
<td>0.6 (0)</td>
</tr>
<tr>
<td></td>
<td>2006</td>
<td>2.8 (3)</td>
<td>1.2 (1)</td>
<td>0.7 (0)</td>
</tr>
<tr>
<td></td>
<td>2007</td>
<td>2.6 (2)</td>
<td>1.3 (1)</td>
<td>0.6 (0)</td>
</tr>
<tr>
<td></td>
<td>2008</td>
<td>2.4 (2)</td>
<td>1.2 (1)</td>
<td>0.5 (0)</td>
</tr>
<tr>
<td>York</td>
<td>2008</td>
<td>2.0 (2)</td>
<td>1.0 (1)</td>
<td>0.5 (0)</td>
</tr>
</tbody>
</table>

Considering the number of programming languages students know well or very well, Figure 4 confirms that almost half of the students from both ETH and York 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 (these numbers include the students that stated being novice programmers). Compared to the ETH students, there are more students at York that state to know only one language well or very well. At ETH there are more students at the other extreme stating to know three or more languages well or very well. These differences have not proven to be statistically significant.

So far, there were no significant differences between the York participants and the ETH participants. Where the difference between ETH and York students starts is in the particular programming languages that they state to know. Visual Basic is the top language at York followed by the scripting languages JavaScript and PHP (see Figure 5). For ETH, the top language is PHP, followed by C++ and C.

Figure 5 shows the 15 programming languages and the percentages of students both at ETH and York 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 *, show significant differences (using the Mann-Whitney test for two independent samples with p < 0.05) between the two populations. The analysis takes into consideration the answers from all students (including programming novices) for the year of 2008.

7. SUMMARY

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

There are several possible factors for this similarity. First, courses on computer science are optional in the high school systems of both countries. This means that there is no regulation that requires schools to offer programming courses. Also, ETH and University of York are both competitive and recruit nationwide. And 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 tough 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 a similar absence of computer science in secondary schools.

It also makes it possible to picture the prototypical CS major taking CS1 at such institutions:

He is between 18 and 20 years old, has learnt one programming language well and has worked with another two languages. He has learnt programming by himself and has a long experience with computers.

At university, he is part of a very diverse student body (and this is unlikely to change in the next years). His fellow students have, at one end, no prior programming experience at all (around 20%) or only moderate knowledge (around 30%). At the other end, his fellow students have expertise in multiple programming languages (around 30% know more than three languages in depth) and may have worked in a job where programming was a substantial part (between 20% and 30%).

8. FUTURE WORK AND CONCLUSIONS

The present study is a first step towards the development of a general picture of entering CS majors’ prior computing and programming expertise.

The current status is based on the data of two European universities located in distinct countries with similar high school regulations concerning computer science. To fully address the issue of generalization, we will continue to query students at other universities in different countries.

The prototypical student already knows one programming language well and two other languages a little. He has learnt his first programming language in self-study and might not have good design principles and algorithmic knowledge, but he feels confident with his skills. His fellow students are very diverse: about 30% are novices or almost novices having only moderate knowledge of programming while another 30% know three or more languages in depth. The challenge, for those who teach introductory programming, is to bridge the gap using innovative teaching strategies, tools and enthusiasm.

9. REFERENCES


