Do Students’ Programming Skills Depend on Programming Language?

Miloš Savić, Mirjana Ivanović, Zoran Budimac and Miloš Radovanović

Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad
Trg D. Obradovića 4, 21 000 Novi Sad, Serbia

Abstract. Bachelor studies in Computer Science at our department in the last decades cover several successive core courses in programming: Introduction to Programming, Data Structures and Algorithms 1 and 2, Operating Systems and Compiler Construction. For a long time our intention was not to insist on the realization of subjects in a specific programming language, but to put emphasis on abstract reasoning and appropriate data structures and algorithms. Also, to avoid teaching different languages and programming environments, we decided to use one good educational language – Modula-2. In the last several years we were under different kinds of pressure to change the language. Starting from the last school-year we decided to adopt Java within the introductory programming course, using the imperative approach first. Some comparisons of students’ advancements and success between Modula-2 and Java generations are presented in the paper. The results of the analytical evaluation indicate that the choice of the first programming language does not have a deep influence to students’ success at the course.

Keywords: Teaching programming, Programming paradigms, First programming language.

PACS: 07.05.Bx, 01.50.-i.

INTRODUCTION

The courses on introduction to programming are part of almost all bachelor studies of Computer Science [4, 6]. For students it is extremely important to gain programming knowledge and skills. The study of Computer Science at our department began about 35 years ago. During the years, studies have been evolving according to different local and global requests and trends in the area [2, 8]. The serious problem we encountered during these innovations was the challenge of choice of the first programming language (PL) and paradigm, as it is important for later development of a programmer. Still, it is a very important dilemma which PL/paradigm and approach to adopt for teaching Introduction to Programming (popularly referred to as CS1).

After many changes in syllabi in the last 20 years, the study of Computer Science at our department still offers essential interconnected programming courses: Introduction to Programming, Data Structures and Algorithms 1 and 2, Operating Systems 1, Compiler Construction. Up to the last school year, our main intention in these courses has been to emphasize abstract structures, and it was decided to choose one good educational language and use it within all these courses. However, in the last years we were under constant pressure (local industry, parents, students, etc.) to change the language for the introductory programming course. Finally, after months of evaluation of different candidates we decided to use Java, applying the imperative-first approach.

The aim of this paper is to try to answer whether the change of the first language influenced students’ programming abilities. We will examine similarities and differences between Modula-2 and Java generations taking the introductory programming course regarding their programming skills and acquired knowledge.

The rest of the paper is organized as follows. The next section discusses significance of programming language for the introductory course. Then, we present an analytical evaluation of the last three generations of students at the course. Conclusions and summary are given in the last section.

SUITABLE PROGRAMMING LANGUAGE FOR THE INTRODUCTORY COURSE

The choice of programming language for CS1 at majority of Universities is still a critical decision [3, 5, 7, 9]. In the long history of teaching CS1, numerous programming languages and paradigms have been used at different universities: Pascal, Ada, Modula-2, C-like languages, Java. For a long time the imperative paradigm was predominant. The direction of historical development has probably not been an accident and programming started
with basic concepts such as the notions of constant, variable, algorithm and procedure, not with the notion of objects.

Having in mind a wide range of requirements, 20 years ago we had to select an appropriate first programming language. We have been assessing and comparing languages like Pascal, Ada, C, and Java. Java seemed to be a good choice, but some authors suggested that Java is not a good first-course programming language [1, 2, 8]. Also, some authors were against the object-first approach as it puts an unnecessary burden on the students. Afterwards we selected Modula-2 as it provides an adequate number of facilities and is suitable to be taught as one “major” programming language throughout the whole computer science education: Data Structures and Algorithms 1 and 2, Operating Systems and Compiler Construction. This educational language allows teachers to concentrate on essential programming concepts (data structures, algorithms, and principles) rather than to teach programming language features.

The Introduction to Programming course at our department lasts one semester with two classes for teaching, two classes for exercising and one class for lab work per week. The aim of the course is to allow students to gain basic programming skills that they will use during further courses. After introducing essential elements of the imperative programming style, the rest of the course is devoted to: iterative and recursive techniques; most effective way of implementation of sums and products; general recurrent sequences; polynomials as an array ADT. The assessment of the course consists of four practical (lab) tests, two theoretical tests and the final (oral) exam. Theoretical tests covering aforementioned topics are not obligatory, but one theoretical test can substitute one (out of a maximum 4) question at the final exam. At the first practical test students have to solve problems involving simple conditional and loop statements. Nested loop statements, arrays and matrices are covered by the second practical test. At the third practical test students need to implement a complex recurrence relation providing two recursive and one iterative solution. The final practical test faces students with problems involving an abstract data type describing polynomials. A student has to achieve at least 30 points (out of maximum 60) at practical tests in order to approach the final exam.

Unfortunately, during last decade we have been heavily exposed to constant pressure of industrial and labor requirements to have employees tailored to their current needs and business practices. Also, we realized that our study programs must be competitive to similar study programs at other faculties. Having all this in mind we finally decided to change first programming language. After all, we were not sure if we did it for marketing purposes or was it a real need?

We are witnesses of rapid changes in the ICT domain and teachers must have in mind constant re-evaluation of the choice of the first programming language and their teaching methodology. As a consequence of this rapid development programmers have to be able to cover many more languages in their lifetime than ever before. Changing education patterns means that students need to be prepared for lifelong learning.

According to above-mentioned facts we again analyzed and evaluated the criteria for the first programming language and decided to switch to Java. After one year of experience in teaching Java within the CS1 course we would like to investigate whether students’ programming skills strongly depend on the introductory programming language. In order to address the research question we will compare students’ scores at practical and theoretical tests for the last three generations of students.

ANALYTICAL EVALUATION

The last three generations of students that took the introductory programming course at our department are denoted as G2012 (the generation of students enrolled in 2012), G2013 and G2014. The same teaching design, assessment and grading methodology were applied for all three generations. The only significant difference between generations is that G2014 learned the basic programming concepts in a procedural subset of Java, while the older generations had the course based on Modula-2. The analysis presented in this section will be focused on students’ score at practical and theoretical tests since students from the Java generation still can approach the final exam in the next two exam periods of this school year.

A student is considered regular if he/she attended all practical and theoretical tests. Table 1 shows the number of enrolled students per generation, the number of regular students, and the average score of regular students at practical and theoretical tests. The average scores at practical and theoretical tests are almost the same, indicating that the choice of the first programming language does not have a profound, systematic influence on students’ success at the course. Indeed, the application of the one-way ANOVA statistical test revealed that differences in average scores are not statistically significant ($F = 7.65 \cdot 10^{-5}, p = 0.99$ for practical tests; $F = 0.33, p = 0.71$ for...
theoretical tests). The homogeneity of variances assumption was checked using the Levene test. The same conclusion is also obtained using the Kruskal-Wallis test which is a non-parametric equivalent of one-way ANOVA ($H = 0.15, p = 0.92$ for practical tests; $H = 0.6, p = 0.74$ for theoretical tests). It is also important to observe that the average score at practical tests is notably higher than the average score at theoretical tests: an average regular student achieved approximately 65% of the maximum score at practical and 50% of the maximum score at theoretical tests.

**TABLE 1.** Basic information about the generations. AVG(P) denotes the average grade at practical tests (the maximum score is 60 points), while AVG(T) denotes the average grade at theoretical tests (the maximum score is 20 points) for regular students.

<table>
<thead>
<tr>
<th>Generation</th>
<th>Total students</th>
<th>Regular students</th>
<th>AVG(P)</th>
<th>AVG(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G2012</td>
<td>78</td>
<td>60 (76.92%)</td>
<td>40.56</td>
<td>10.38</td>
</tr>
<tr>
<td>G2013</td>
<td>97</td>
<td>60 (61.85%)</td>
<td>40.58</td>
<td>10.36</td>
</tr>
<tr>
<td>G2014</td>
<td>82</td>
<td>57 (65.51%)</td>
<td>40.36</td>
<td>9.78</td>
</tr>
</tbody>
</table>

We classified students according to their score at practical/theoretical tests into three categories:

- **B** category - students that achieved a bad score which is less than 50% of the maximum score,
- **G** category - students that achieved a good score which is equal or higher than 50%, but less than 85%, of the maximum score, and
- **E** category - students that achieved a very good score which is higher than or equal to 85% of the maximum score.

Tables 2 and 3 summarize results of the classification for practical and theoretical tests, respectively, showing the percentage of students and the average score in each category. It can be seen that G2014 has a smaller percentage of **E** students at both practical and theoretical tests compared to the previous generations. This means that the best students were slightly more frequent in the Modula-2 generations. Regarding practical tests, the most dominant category of students in all three generations are students achieving good scores. The fraction of G2014 students belonging to the **G** category at theoretical test is approximately the same as the fractions of **G** students from older generations. On the other side, G2014 has the highest percentage of students that achieved a bad score at theoretical tests.

**TABLE 2.** The results of the classification of students according to the score at practical tests.

<table>
<thead>
<tr>
<th>Class</th>
<th>G2012 (%)</th>
<th>AVG(G2012)</th>
<th>G2013 (%)</th>
<th>AVG(G2013)</th>
<th>G2014 (%)</th>
<th>AVG(G2014)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>13.3</td>
<td>14.1</td>
<td>25</td>
<td>21.7</td>
<td>14</td>
<td>21.8</td>
</tr>
<tr>
<td>G</td>
<td>55</td>
<td>38</td>
<td>43.3</td>
<td>39.7</td>
<td>63.2</td>
<td>38.7</td>
</tr>
<tr>
<td>E</td>
<td>31.7</td>
<td>56.2</td>
<td>31.7</td>
<td>56.7</td>
<td>22.8</td>
<td>56.4</td>
</tr>
</tbody>
</table>

**TABLE 3.** The results of the classification of students according to the score at theoretical tests.

<table>
<thead>
<tr>
<th>Class</th>
<th>G2012 (%)</th>
<th>AVG(G2012)</th>
<th>G2013 (%)</th>
<th>AVG(G2013)</th>
<th>G2014 (%)</th>
<th>AVG(G2014)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>43.3</td>
<td>6.1</td>
<td>45</td>
<td>6.7</td>
<td>50.9</td>
<td>6.5</td>
</tr>
<tr>
<td>G</td>
<td>48.3</td>
<td>12.9</td>
<td>46.7</td>
<td>12.5</td>
<td>45.6</td>
<td>12.9</td>
</tr>
<tr>
<td>E</td>
<td>8.3</td>
<td>17.8</td>
<td>8.3</td>
<td>18.4</td>
<td>3.5</td>
<td>18</td>
</tr>
</tbody>
</table>

Figure 1 shows the cross-classification of students by scores at practical and theoretical tests. Categories are marked by $X-Y$, where $X$ is the classification class, while $Y$ denotes the type of test ($P$ - practical, $T$ - theoretical). It can be observed that in all three generations students with a very good score at theoretical tests have at the same time a very good score at practical tests. On the other hand, each generation contains a significant fraction of regular students (around 20%) that achieved an excellent score at practical tests (the **E-P** category), but performed slightly worse at theoretical tests (the **T-G** category). Furthermore, there are two students in G2012 with an excellent score at practical tests who had a bad score at theoretical tests. Interestingly, both of them passed the exam with the second highest grade in the first exam period.

**FIGURE 1.** Cross-classification tables of scores at theoretical and practical tests for regular students. The number of students for all categories is given in percentages.
A significant fraction of students in all three generations (from 23% in G2013 to 36.8% in G2014) achieved a good score at practical tests, but had a bad score at theoretical tests. Those students obtained 36 points at practical tests on average (60% of the maximum score) and 6 points at theoretical tests (30% of the maximum score). Finally, it can be seen that students performing badly at practical tests also tend to have a bad score at theoretical tests.

CONCLUSIONS AND FUTURE WORK

It seems that choice of the first programming language (i.e. a “mother-tongue”) for computer science education is nowadays a more responsible task than ever before. One of the reasons is the existence of many programming paradigms and languages and most of them compete for widely used in industry and real world applications. Still, a lot of debates concentrate on this choice: should we teach C, Java, Net, Python, Perl, Scala, or JavaScript? Education evidently needs to follow these trends, but also has to satisfy other (methodological, didactical, etc.) parameters. All programming languages and paradigms evidently have some advantages as well as disadvantages, making the decision more difficult.

In this paper we examined students’ scores at practical and theoretical tests for the last three generations of students of the introductory programming course at our department. The only significant difference between examined generations is that the last one had the course based on Java, while the other two learned basic programming concepts relying on Modula-2. The evaluation showed that average grades at practical/theoretical tests of students that regularly attended the tests are nearly the same. The absence of statistically significant differences between generations was also verified using ANOVA statistical tests. This result suggests that the choice of the first programming language does not have a profound influence to students’ success at the course. We also classified students according to their scores at practical/theoretical tests into three categories (bad, good and very good), observing that the generations possess similar characteristics regarding classification outcomes and cross-tabulation of categories. In our future work, when data becomes available, we plan to extend the current analysis with other important factors such as the final grade at the course, average grade in high school, and grades at other university courses.

ACKNOWLEDGMENTS

This paper is part of the research project Infrastructure for Technology Enhanced Learning in Serbia supported by the Ministry of Education, Science and Technological Development Republic of Serbia [Project No. 47003].

REFERENCES