PREDICTORS OF NOVICES PROGRAMMERS' PERFORMANCE

Nikolina Bubica¹, Ivica Boljat²

¹ Elementary School of Mokosica (CROATIA)
² Faculty of Science, Split (CROATIA)

Abstract

Despite the awareness of the important role of programming in today's technological lifestyle and the fact that programming jobs are among the most sought after professions today, computer science students consider programming contents overly demanding and often avoid them.

This paper aims to provide an overview of the relevant literature and studies that examine factors that may significantly affect the performance of programming. We will give an overview of good and bad predictors of success in programming and factors that, so far, have not shown the predictions' strength or their impact on the performance of programming hasn't been sufficiently explored.

Knowledge of these factors is extremely important for computer science teachers in order to apply the appropriate teaching model for novices that emphasizes the development of the ability to anticipate and understand what happens when a specific computer program performs. Factors like mathematical knowledge, attribution of success, good spatial maps sketching have proved to be good predictors of programming success while surface learning approach showed strong negative impact on novices' programming performance. On the other hand, despite the expectations, student's gender, number of years of programming, number of familiar programming languages, haven't show any significant impact on programming performance.

Keywords: programming, novices, predictors, teaching.

1 INTRODUCTION

Teaching programming is an integral part of number of introductory computer science courses curriculum. Although, we live in a world of technology where students use at least one digital device and a multitude of software applications regularly, it looks like programming contents are not particularly attractive to students. Indeed, they find it demanding and unattractive. Most researches of teaching programming talk about low introductory programming pass-rates and poor passing average results. Student's results on these tests could be considered as evidence of complexity and weight of programming content. Students other subjects scores were significantly better than their introductory programming scores. On the other hand, it seems reasonable to believe that with careful instruction, proper motivation and plenty of time an individual can learn everything he wants. Some teachers believe that this is not true for math, some for art and music, and programming teachers point out that this is definitely not true for programming. Why is programming so demanding? Which factors influence the most success in programming? Although there are greater number of studies that have attempted to answer this question, we still haven't discovered factors that could be considered extremely powerful predictors of success in programming. We can talk about the factors that have shown a strong, positive or negative impact on the programming performance, but also we could talk of factors that despite expectations didn’t express the power of a programming predictor. One of the reasons why there are still no stronger results in the field of predictors of programming could be found in the fact that subject of programming is not widespread, it is taught mainly at universities, and in some primary and secondary schools. On the other hand, the look at the most daily advertised jobs in many countries confirms the continuing industry need for computer experts especially developers. It is the computer industry that increasingly emphasizes the importance of recognizing predictors of programming in order to promptly focus to those students who are predisposed to achieve success in this field.
2 FACTORS THAT CAN INFLUENCE SUCCESS IN PROGRAMMING

2.1 What do we consider success in programming?

Studying the performance of learning to programming has been one of the fundamental issues of computer science research for many years. What influences the students that some easily acquire programming concepts while others are extremely troubling with same concepts? While at beginning, teachers were eager to find in their classes those rare students who had that "something specially" needed for mastering programming, today many teachers wonder why their even "capable" students do not want to deal with programming concepts. In most researches success in programming is viewed through the introductory programming courses, which are implemented in different names, different volume of content covered by a particular subject and through different levels of difficulty of content being implemented at various educational institutions. Because of these differences it is difficult to talk about the compliance of each introductory subject, and the result of research as well. Because in most researches success in introductory programming is viewed through a variety of quantitative measures such as the withdrawal rate of the subject, the permanence of the subject, students’ performance in exams during subject, and final exam. Student's achievement can also be viewed from the teacher's perspective. Using different qualitative methods we can investigate the understanding that is applied to content being taught, understanding of certain qualitative students' characteristics, understanding the connection between success in programming with previous experience of students, we can emphasize student's behavior and attitudes, and finally understand the role of teachers in students development [1].

2.2 Standard factors considered to influence programming

The number of studies investigating factors important for success in general is great. The best indicators of success in all disciplines are considered to be self-provided success, one's attitude, his enthusiasm with certain content, and one's general academic motivation. These factors really have a strong impact on the performance of students, but do not separate programming from other disciplines. To find out the specific indicators of success in programming it was necessary to conduct more sensitive researches in the area of programming performance. Various attempts to predict success in programming usually began with intuitive assumptions of researchers themselves. One of the first good prerequisites for success in programming imposed familiarity with the area from which the task of programming came from [2]. However, further research into the process of designing software showed that just insufficient knowledge of a particular area could facilitate the programming of specific tasks as it provided a critical review to the problem [3]. The individual is sufficiently beyond the problems so that he can spot all its specificity, and thus make better software solution. Today this approach is often used in the planning of tasks programming teams [4].

In previous researches, which aimed to find some factors that might predict success in programming were usually observed factors such as expertise in their spoken language, the number of used and analyzed programming languages, mathematical ability, musical ability, previous academic degree, measures of general intelligence, self-confidence of students and so. Among all listed as the most powerful predictors of stressed precisely the degree of comfort, a mathematical ability and attributions of success dependent on luck [5]. Interestingly, it showed a significant positive correlation between the marks at the end of programming objects and spatial map sketching. Students who were able to create a spatial map containing all known objects and sequences of locations achieved better results in programming of those who were able to draw only the direction of motion [6, 7, 8]. It has shown that different navigational strategies could positively affect the quality of programming code and thus form a conceptualization of previous computer experience.

The study of gender as a factor that can significantly affect the performance of programming arises almost naturally given to fact that the world of programming is dominated by male persons. Despite expectations that gender is an important factor, several studies have confirmed the exact opposite. That gender is not factor that affects the performance of programming [9, 10, 11, 12]. However, it has been shown that women are rather than men to give up some more technical content of computer science like computer programming [13]. Excellent mathematical knowledge was confirmed as a strong predictor of success in programming through a number of studies [9, 14, 15]. Further exploration of mathematical knowledge as a predictor of success in programming has shown that good adoption of discrete mathematics and mathematical calculus could be considered a good predictor of success in programming [12].
Students of introductory programming courses are often with different computer knowledge and experience. It seems natural to expect that students with less prior experience in using computers are likely to be less confident in the initial programming tasks and have lower expectations of success. Studies of administered student’s learning styles for example Colb learning styles and styles according to Gregorc haven’t offered unique results up to now. According to Kolb, the various learning styles are suitable for specific learning environments including learning how to program, in which the student is trying to solve a problem, apply a variety of skills, understand and identify relationships between concepts. Examining the impact of Colb learning styles gave no strong conclusion about influence on programming but it showed that students who chose the subject of programming predominantly cultivated Colb’s convergent learning style and achieved better results compared to other students [9]. On the other hand, the investigation of the influence of learning styles according to the Gregorc on programming are lower in number, but a strong positive influence on the results of the programming for abstract/random dimension indicated that a learning styles according to Gregorc could be a good predictor of the success of programming [8]. Because of the inconsistency of results concerning the impact of learning styles on programming performance a meta-analysis of existing researches might lead to a valid conclusion about the impact of learning styles on success in learning programming.

How do novice developers adopt tough concepts in introductory programming courses? In early seventies phenomenographic research demonstrated two different approaches to learning that students usually used. It identified deep approach to learning in which students tend to develop a real understanding of what they learn and surface approach to learning in which students just want to do a task that they got form their teachers. The studies that have been studying the learning strategies showed that the deep approach to learning is positively correlated with the grade of the initial programming while a surface approach to learning suggests a negative correlation with the same grade [6].

What strategies beginners apply to solve problems that are based on difficult concepts? We know that there is a strong correlation of success in programming with the ability to solve problems in other sciences such as mathematics, physics and so [11]. Although it was shown that pre-computing experience is not important for achieving success in programming, very interesting results related to the self-assessment of the performance offered by the research of student comments and observations were collected during the programming task. Positive previous programming experience strongly support a positive self-assessment of student’s work [16] which is generally regarded as a positive factor for any activity.

Adoption of programming skills requires of student to design an efficient computer model and the construction of completely new cognitive model through which student explains how the computer works, anticipates and understands what happens when performing a specific computer program [17]. Learning to program involves the creation of viable mental models of basic programming concepts. In teaching programming for teachers is very important to recognize the mental models of basic programming concepts that students possess, for those students who create viable mental models achieve significantly better results in solving the programming tasks than those with non-viable mental models [18, 19, 20].

2.3 Programming Languages and Experience – how much do they matter?

One of the interesting research questions of computer science certainly is a choice of the first programming language for beginners. The choice of the first language used in introductory programming is often the cause of the protracted and sometimes highly emotional debate. Currently, there are approximately 8000 known programming languages that are documented on the World Wide Web [21]. Which of these languages should acquaint students in the best way with the appropriate programming concepts and keep students’ interest towards programming as an essential part of computer science? Is the choice of the first programming language one of the factors that could significantly determine the success of students in programming? Research of the success of novice programmers indicates that the first programming language strongly influences success in programming [11]. Despite the large number of existing programming languages, some programming languages are extremely popular, while others simply disappear long before they could become famous. Analysis of the adoption of programming languages has shown that in the actual use there is a small number of programming languages in the actual use, and that despite the existing quality analysis of suitable languages for teaching the selection of programming language is usually influenced by factors such as the existence of open source libraries, the existence of the finished code for learning and previous programming experience and factors like the accessibility of semantics,
reliability and features of the programming language itself become less important [22]. Because of the strong impact of the first programming language on novice’s success teachers should devote attention to this matter.

In contrast to the importance of the election of the first programming languages other matters related to the programming languages so far have not shown a greater impact on success in programming. Thus, the number of introduced programming languages [8], the total number of years in programming [8] and previous programming experience [9, 10, 11] are factors that have so far not shown significant performance for programming. Although it seems surprising that several researches have confirmed that prior programming experience is not an important factor for achieving success in introductory programming courses but we couldn’t say that the existence of programming experience is completely irrelevant. The recent study of the impact of previous programming experience on success in programming showed that students with previous experience of programming performed better than those without previous experience [23]. Also, students may have a negative feeling about their effectiveness in programming even after their own positive experiences in programming tasks, but also that after the negative programming experience student can have a positive feeling about his effectiveness [16]. Although it is naturally expected that the positive programming experiences leads to the positive sense of self efficacy cause of research results may lie in some other factors in the introductory programming objects such as the choice of the first programming languages, teaching strategies, students motivation, difficulty of teaching content but also among the factors associated with the teachers....

While working with novice programmers it is extremely important for a teacher to know the way how novices adopt concepts they are being taught to. Experienced programmer creates code to solve the problem by applying strategies that are derived from experience of solving the problems from the past, which is not the case for novice programmers [24, 25]. They can very well understand the syntax rules of the programming language and write simple programs, but it is often difficult for them to find solutions to problems if they don’t have the experience of an expert. Knowing the difference between the ways in which novices and experts analyze and solve problems could help teachers. Experts create mental representations that are more than the sum of the information obtained by reading the program. They create flexible and navigation mental presentations what make us think that the way to teach novice programmers can be a predictor of their success. Novice programmers build knowledge by constantly filling out and connecting pieces of knowledge which usually is a lengthy process. It takes about 10 years to become proficient novice programmer [26]. In recent years, it is considered that an intuitive teacher could get a programmer-level understanding of the program within two to four years with an emphasis on the need for extremely patient work with students [27].

Just knowing and understanding the difference between beginners and experts will emphasize concepts that beginners are focused on. A detailed comparison of expert and novice expressed some newer and confirmed some old "positive" and "negative" predictors of programming [7]. As good predictor of programming were confirmed the good sketching spatial map [6, 7, 28] the ability to induce (to observe the rules in the example), the use of sustainable mental models [7, 20, 25], sense of humor, patience and perseverance, skill of manipulating the rules of grammar and syntax. If as negative predictors we consider the factors showing negative correlation with success in the performance of programming in such factors we can include: increasing number of hours spent in playing computer games, frequent frustration with the rapid withdrawal, an aversion to programming, a sense of failure when programming, the use of different programming models to solve the problem, the belief that the best solution is to memorize in order to learn, the belief that real programmers immediately see the problem, the belief that it takes a lot of knowledge about programming to achieve success [7]. The issue of the impact of computer games is very interesting, but insufficiently explored with regard to the question of types of computer games played. There are indications that successfully playing certain thinking games for ex. MasterMind may indicate the future successful programming performance [29].

2.4 Non-traditional approach to the study of programming predictors

A greater number of studies carried out so far, exploring the success in programming were realized by a variety of statistical tests, which often were not able to retrieve the actual change in the student’s progress in learning. Unlike statistical tests, the research of programming, predictors based on the core aspects of the student's behavior, while they are programming, defined new factors that could affect students’ progress in learning that occurs over time [8]. Such predictors were able to dynamically identify students with learning difficulties, and could be used to create an expert system.
appropriate to respond to students when the need arises. The results revealed a strong positive correlation with good success in programming specifically for students who had a high percentage of successful consecutive program compilation. Students with poor success in programming had a high percentage of consecutive errors in programming exercises and a greater number of hours spent on debugging their program [8].

Teachers of introductory programming courses are usually very interested in early identification of those students who can easily adopt the content of programming, but also those students who even at the outset are faced with many difficulties. It has been already shown that in the first weeks of the subject teacher can use specific questions to identify best students [23]. The implementation of a written test consisting of specific predictive questions at the same beginning of the course (Fine Grain Clicker Exam) showed that just the students who achieved the best results on that early valuation achieved much better results than other students throughout the course and on the final course exam. The result of this research is a trace of the idea of creating an universal test that would be able to predict the tendency towards programming even before the student begins the introductory programming course. Similar path took researchers of the Center for Autism Research (Autism Research Centre, Cambridge). The study of possible predictors of programming was initiated by researchers who studied whether there was a relationship between the "systemizing Quotient" (SQ) and "empathy quotient" (EQ) with success in programming. Systemizing quotient (SQ) reflects the ease with which an individual understands the systems of objects while the empathy quotient (EQ) reflects the ease by which an individual understands the emotions of the people (EQ). People whose value of the SQ is much lower than the EQ value like interacting with other people in everyday life, at the same time they consider machinery and computers mysterious, cold and heartless. People whose value SQ is much higher than the EQ value prefer to deal with the ordered systems of buildings in everyday life, they intuitively understand them better than human relationships that they consider confusing. Research showed that linking programming and individual values of EQ and SQ showed no correlation, while the difference of values SQ-EQ showed high correlation with success in programming or we can say that there is great strength prediction bias toward programming [30].

Mastering the programming language requires not only some initial student capabilities, but also a tendency that the person engages extremely and invests great effort what person with high difference of SQ-EQ finds appealing and satisfying. People with low values of SQ-EQ such engagement consider uninteresting and less useful than some other activities that involve more interaction with humans. This is consistent with the results of several studies dealing with the discovery of personal characteristics like cognitive abilities, value attitudes, personalities and successful solution of the tasks of programming influencing the programming performance. Personality of the individual [31], his cognitive abilities [32] and belief in the theoretical value [31] have shown the strength of predictors of programming. A person with a strong belief in the theoretical value is likely to be successful in the programming tasks as well as a person with a strong personality (self-esteem, self-efficacy, locus of control, high and low level of neurotism) [31]. Although it is often considered that cognitive ability is sufficiently strong predictor of programming it should be borne in mind that a long-term challenging activity such as programming is not only dependent on cognitive abilities, but also on some personal factors that could make individuals with lower cognitive abilities to invest greater effort and outperform those with higher cognitive abilities but lower susceptibility to the performance. People with a strong belief in the theory are highly focused and objective, interested to work independently, always happy to look for a reason and cause of the existence of an object and events [31] what is often associated with programming and programmers.

Can the ability of a person to achieve success in programming be identified in advance even before the person starts programming? An entirely new direction in the study of characteristics and factors important for programming performance represents the use of functional magnetic resonance imaging, which observes activities that occur in parts of the human brain while programming and understanding programs. The research revealed activities in five different areas of the human brain that are associated with working memory, attention and language processing [33] while the act of programming. Processing of the language is very important for understanding the work of a computer program which confirms Dijkstra's claim [34] that the exercising language skills, in addition to problem-solving skills is important for effective learning of programming.
3 TEACHERS - HOW IMPORTANT ARE THEY IN TEACHING INTRODUCTORY PROGRAMMING COURSES?

Most of the observed factors are part of personal epistemology computer science students. The teacher can hardly significantly impact on most of those factors, but it is extremely important for them to know well their students. In that way teachers can better meet the needs of their students and facilitate their adoption of challenging programming concepts. There is a growing number of studies which talk about the effectiveness of new approaches, methodologies and learning environments, but often neglect the importance of the teachers’ quality and motivation and how precisely this factor affected the performance of an explored method. To what extent can the teacher's motivation and enthusiasm contribute to the success of some methods and approaches? The human factor is a key factor in the educational system [35]. The experience of successful schools point out that getting the right people to become teachers, developing them to effective educators and ensuring that the system offers the best possible instruction for every student is critical to education quality [36, 37]. The research of McCracken’s group firstly indicated how teachers were purely familiar with the actual understanding of the content that was going on in students mind and encouraged further research in this direction [38]. Newer phenomenographic international research among teachers conducted at several universities has shown that researching teachers could offer a multitude of responses on issues of student's success in programming and that the most successful among teachers were those who cherished the developmental viewpoint, teachers who included and used all levels of understanding to create interaction between teachers and students [1]. Such teacher uses a variety of approaches to teaching, uses educational materials adapted to different students learning styles, offers students appropriate examples and problems to work effectively, monitors and evaluates students, and simply such teacher creates a positive learning environment in his classroom.

In recent years, more and more attention is paid to the factor of motivation and engagement of students in computer science and are explored the various forms of student incentives. The prize in the competition or extra points on the exam due to the success of a programming event has shown the ability to motivate students in introductory programming courses [39, 40, 41]. Such strategies have proven to be extremely motivating for students, and the mere existence of competition students evaluated very positively. The biggest challenge for teachers of computer science lies in working with a diverse group of students who often come with a variety of prior knowledge. For teacher it is very difficult to find an appropriate level of difficulty of teaching facilities for all students. If the level of presentation is too low, some of the best students will be bored and will feel demotivated to work. Some teachers advocate for new work methodology which replaces the final exam with a series of activities in the context of continuous assessment and the use of online programming competition in which all students take part. Approach has proven to be extremely effective, the pass rates of introductory programming courses almost doubled [42]. However, in the absence of a large number of studies that aim to reveal the impact of competition on student's success in programming these results should be viewed broadly.

4 CONCLUSION

The prediction of student's success in an introductory programming course is the constant theme of researches in teaching computer science. In this paper we offered an overview of the factors that through various studies have shown the power of one predictor of programming performance. We also discussed the factors that despite expectations have not shown to be essential for the performance of programming or such factors need to be further explored with some other methodologies. Overview of the analyzed study results is shown in table Fig.1.

From the survey results it is clear that we will not soon discover a magical factor that could accurately predict a student's success in programming. For researchers is more obvious that the answer to this question lies in researching combination of several factors on the performance of novice programmers, but also in some other, so far neglected factors such as factors related to the teachers.
<table>
<thead>
<tr>
<th>Potential factors of programming success</th>
<th>Positive correlation</th>
<th>Negative correlation</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>mathematical knowledge</strong></td>
<td>Byrne &amp; Lyons, 2001</td>
<td>Wilson &amp; Shrock, 2001</td>
<td>Stein, 2002</td>
</tr>
<tr>
<td><strong>spatial map sketching</strong></td>
<td>Simon &amp; others, 2006</td>
<td>Kranch, 2011</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Watson &amp; others, 2014</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>sense of comfort</strong></td>
<td>Bergin &amp; others, 2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>attribution of success</strong></td>
<td>Wilson, Shrock, 2001</td>
<td>Henry &amp; others, 1994</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ventura, 2005</td>
<td>Kinnunen &amp; Simon, 2014</td>
<td></td>
</tr>
<tr>
<td><strong>Learning styles</strong></td>
<td>Kolb learning style</td>
<td>Pillay, Jugoo, 2005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gregorc learning style</td>
<td>Watson &amp; others, 2014</td>
<td></td>
</tr>
<tr>
<td><strong>first programming languages</strong></td>
<td>Pillay, Jugoo 2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Students’ behavior while programming</strong></td>
<td>high percentage of successful consecutive program compilations</td>
<td>Watson &amp; others, 2014</td>
<td></td>
</tr>
<tr>
<td><strong>Learning strategies</strong></td>
<td>deep approach</td>
<td>Simon, Robins &amp; others, 2006</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simon, Cuts &amp; others, 2006</td>
<td></td>
</tr>
<tr>
<td></td>
<td>surface approach</td>
<td>Simon &amp; others, 2006</td>
<td></td>
</tr>
<tr>
<td><strong>gender</strong></td>
<td></td>
<td>Byrne, 2001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chumra, 1998</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pillay, Jugoo (Pillay, Nelishia; Jugoo, Vikash R., 2005)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pioro, 2006</td>
<td></td>
</tr>
<tr>
<td><strong>previous programming experience</strong></td>
<td>Sheard &amp; others, 2008</td>
<td>Kinnunen &amp; Simon, 2014</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Porter, Zingaro, Lister, 2014</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>total number of years spend in programming</strong></td>
<td></td>
<td>Watson &amp; others, 2014</td>
<td></td>
</tr>
<tr>
<td>Factor</td>
<td>References</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of introduced programming languages</td>
<td>Watson &amp; others, 2014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>using viable mental models</td>
<td>Bornat &amp; others, 2008; Kranch, 2011; Kranch, 2012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>solving problem abilities</td>
<td>Pillay, Jugoo, 2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>greater number spent in playing computer games</td>
<td>Lorenzen &amp; Chang, 2006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>exercising languages skills</td>
<td>Siegmund &amp; others, 2014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>English as native language</td>
<td>Sheard &amp; others, 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>students interest and their course expectations</td>
<td>Sheard &amp; others, 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQ-EQ value</td>
<td>Wray, 2007</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Overview of the analyzed study results concerning programming predictors.

REFERENCES


[6] Simon; Fincher, S.; Robins, A.; Baker, Bob; Box, Ilona; Cutts, Quintin; Raadt, Michael de; Haden, Patricia; Hamer, John; Hamilton, Margaret; Lister, Raymond; Petre, Marian; Sutton, Ken; Tolhurst, Denise; Tutty, Jodi, “Predictors of Success in a First Programming Course,” Australian Computing Education Conference (ACE), vol. 52, pp. 189-196, 2006.


Siegmund, Janet; Kästner, Christian; Apel, Sven; Parnin, Chris; Bethmann, Anja; Leich, Thomas; Saake, Gunter; Brechmann, André, “Understanding Understanding Source Code with Functional Magnetic Resonance Imaging,” in ICSE ’14, May 31 – June 7, Hyderabad, India, 2014.


