Indian Journal of Science and Technology, *Vol 9(46), DOI: 10.17485/ijst/2016/v9i46/107572, December 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Strategies for Increasing the Academic Performance of Students at School-Level Computer Programming Lessons

Spartak Razakhovich Sakibayev* and Bela Razakhovna Sakibayeva

Department of Physics and Mathematics, Zhetysu State University, Taldykorgan, Kazakhstan; spartakrz2000@gmail.com

Abstract

Background/Objectives: This article contains the results of the research conducted by its authors in order to propose the ways enabling school students to demonstrate high academic performance at programming lessons. **Methods**: The authors identify the factors which prevent the advancement and development in the sphere of school programming education. They also formulate several hypotheses aimed at increasing the effectiveness of school programming classes. **Findings**: The authors have found optimal ways of increasing the effectiveness and value of the school-level programming education. All theoretical findings are verified by real-life statistical data obtained from studies of existing teaching practices. **Improvements**: The article provides methodological and practical guidelines aimed at raising the educational value of school programming classes and increasing the students' interest in the subject. It can be used by programming teachers to develop a long-term strategy for ensuring the quality of learning environment and academic performance.

Keywords: Innovations in Education, New Programming Teaching Methods, Programming Education

1. Introduction

Computer programming constitutes a fundamental part of a school computer science curriculum. It develops logical thinking, abstract reasoning, creative skills and other abilities necessary for academic success. Also it is the sphere of human activity which serves as a driving force behind the development and advancement of the modern information technology-based society. Although the training of future competent software developers is the responsibility of higher education institutions, schools can also contribute effectively to this process. It is at the school level that students get their first experience of programming and obtain the most essential programming knowledge and skills. And in many cases namely this school experience influences a student's future attitude towards programming and affects his future academic performance at university level. So it becomes of a great importance to make sure that school programming course is oriented towards bringing an actual educational value and preparation of students for the subsequent academic achievements in the context of software engineering-related disciplines from the university-level computer science curriculum.

Despite the self-evident importance of schools in the preparation of future software developers, existing teaching strategies in the sphere of programming still suffer from the lack of effectiveness and adaptation to permanent changes in the information technology landscape. The existing strategies are still based on traditional approaches and views which decrease the attractiveness of programming classes and cause the formation of negative attitude towards programming among students. The traditionalists use deprecated programming languages and tools and supply students with problems and tasks that have neither educational impact nor the ability to awake the students' interest. It is safe to say that traditional programming teaching strategy is most resistant to changes

**Author for correspondence*

among the disciplines of computer science curriculum and develops in complete isolation from the actual demands and challenges of the modern information technologies.

The main goal of this article is to study the negative factors and practices which prevent existing school programming strategies from imposing a positive educational impact and then offer a new alternative approach towards teaching programming which will increase the usefulness and attractiveness of programming classes in particular and the whole computer science curriculum in general. The authors discuss the disadvantages of the tools, programming languages and problems used within the context of traditional teaching methods and then offer their alternatives which are more suitable to the requirements of modern information infrastructure. Also the authors share their suggestions concerning creation of a unified programming language for the usage exclusively within the context of a school programming class. The theoretical findings of this article are supported by the actual experimental results obtained by the authors during their teaching practice.

2. Methods

This research was initiated by the authors' attempts to find a rational explanation to the fact that so many high-school graduates come to universities so unprepared for academic success in programming-related disciplines. Though a big percentage of these graduates demonstrate a highly-developed capacity for logical and abstract thinking and inclination for mathematical sciences. The first step of the research was to collect data concerning existing practices and strategies of school-level programming education and the academic performance of students within the context of programming classes. The authors studied what programming languages and tools are used in programming lessons and what typical problems and tasks are given to students during these lessons. Also it was necessary to study how the existing practices affect the students' academic performance and their overall attitude towards the subject. The experimental data was collected from local municipal schools.

During the research the authors discovered six main factors which prevent the advancement and development in the sphere of school programming education. These factors are discussed below:

The first factor is the irrelevant platform on which the students work. Traditional teaching approach is that all programming work must be done with the help of some desktop-based (IDE). But the majority of today's students are accustomed to getting various computer services through the help of a favorite Web browser. As such, these desktop IDEs contradict the students' perception of modern computer technologies as "Internet is everything" and so do not contribute to forming a positive attitude towards the lessons.

The second factor is the inevitable logical consequence of the first one and lies in the fact that students can't access their programming tools through their favorite gadgets such as smart phones. Since the prevailing programming tools used in a class are desktop-based, it becomes impossible for students to work within the environment of their favorite gadgets.

The third factor is the overall complexity of programming languages used within the classroom. Traditional approach is that students are asked to use such languages as Pascal and Basic. These are full-featured programming languages which were used decades ago on a large scale for building commercial applications. And as such their functionality and features exceed the needs of problems and tasks commonly given to students at schools.

The fourth factor is related to the first and the third ones and lies in the fact that students are asked to use full-featured programming environments for simple tasks and problems which they are assigned to solve in a classroom. These environments usually provide very complicated user interfaces based on complex menu systems which have a negative effect on students' perception.

The fifth factor is the usage of deprecated programming tools in a classroom. Commonly used languages such as Pascal and Basic are generally considered to be "out-of-date" and thus do not attract students.

The sixth factor is the irrelevance of programming problems commonly assigned to students. Traditional approach is that students are mostly assigned to solve problems of elementary mathematical nature, such as solving simple equations and drawing simple graphs. But the majority of students are attracted by doing Web-related work. They do not find it interesting and stimulating to write desktop and command-line programs of mathematical nature when World Wide Web (WWW) has become the dominant platform.

To eliminate these factors and to increase the attractiveness of programming classes the authors proposed five hypotheses which were tested during the research. These hypotheses are discussed below:

H1: Programming tasks in a classroom should be done using Web-based IDEs. Web has become a dominant platform and there are many areas in which traditional desktop applications are replaced with Web-based applications. Due to their nature, Web-based programming environments provide a simple and minimalist user interface and implement only the most essential features. They are not subject to miscellaneous problems related to installation and deployment. Because of this they are easier to use as they allow student to concentrate on the programming task rather than making him navigate through complex menu system and toolbars of a traditional desktop IDE. From the students' perception Internet support is one of the most important criteria of the attractiveness of a school discipline. Web-based IDEs can be easily accessed from the students' favorite gadgets such as smart phones. Besides according to[1], 92% of teachers say the internet has a "major impact" on their ability to access content, resources, and materials for their teaching.

H2: Client-side web-scripting languages such as JavaScript are an effective replacement to traditional languages, such as Pascal and Basic, for use in a classroom. Students view these traditional programming languages as deprecated since they have no relation to modern Web-driven information technologies and produce output in the form of either desktop or command-line programs. But programming teachers keep using them as they think that only they are well suited for solving typical classroom problems. Surely, there are no strong reasons for such limited views. JavaScript, for example, is a well-featured programming language whose functionality is enough for solving typical classroom problems. It is interpreted rather than compiled. It does not require a separate installation as it is an integral part of all major browsers[2] including those that run on smart phones. JavaScript's programs are accessed through browser, have a direct relation to World Wide Web and thus contribute to forming a positive attitude towards the subject among students.

H3: If the hypotheses H1 and H2 are correct then programming classes should use smart phones instead of desktop computers for the students' programming tasks. Smart phones can easily run JavaScript-written programs and they are the preferred devices owned by the majority of students[3]. In[4] writes that "M-Learning is one of the important and new methodologies adopted by the educators as an effective aspect of providing learning". An ability to do classroom programming tasks directly from within their smart phone's environment increases the attractiveness of a computer class and stimulates an interest in the subject. From the point of view of a school management, smart phones allow schools to save money on buying traditional desktop-based computer classes. In[5] draws attention to the fact that "handheld mobile devices and cellular services are significantly less expensive than PCs and laptops with fixed Internet service". According to[6], "Mobile devices in general and Android-based devices in particular have a potential of becoming a dominating platform for use in programming classes". Also[7] assert that "mobile learning can be used as a hook to re-engage disaffected youth". In[8] states that mobile technology allows everyone to magnify his or her knowledge in every setting". The same concept was stated in[9] where the authors remind that "the main objective of m-learning is to provide the learner the ability to assimilate learning anywhere and at any time". According to[10], "it is inevitable that the educational environment will realize that using these mobile devices on campus would enrich the learning experience".

H4: Typical programming problems and tasks of elementary mathematical nature should not be assigned to students at all. Such problems and tasks usually do not attract students since they have lots of them in mathematical lessons. Instead it is better to give those problems and tasks which contain the elements of a game and yet stimulate logical reasoning[11]. For example, it is effective from an educational point of view to give chess-related problems, such as, for instance, placing the N chess queens[2]. Chess-related problems and tasks are one of the most efficient ways to develop mathematical and algorithmic abilities.

H5: Programming classes need the creation of a fundamentally new programming language designed specifically for the use in a school environment. According to the hypothesis H2 JavaScript is the optimal choice for use in a school programming lesson. But the functionality of this language largely exceeds the needs of a typical school problem or a task. So for the purpose of advancement of school-level computer programming education there arises the need for a new programming language whose functionality strictly adheres to the content of school programming tasks. This new language will implement only the most basic and essential features enough for solving any school-level problem. Also it should be interpreted rather than compiled and run within the browser environment like JavaScript. The desired functionality of the language is given below.

- Language supports only integer and floating-point data types and their arrays[12].
- Language implements the IF statement.
- Language implements only *FOR* loops.
- Input data are read from browser's Web console[13].
- Output data are written to browser's Web console[13].
- Library contains only the most essential functions for mathematical calculations and drawing graphs.
- The language's environment should provide a function for sending a program to a student's email address.

3. Results

To test the formulated hypotheses the authors spent two months accumulating and analyzing experimental data obtained from visiting programming classes in five of the local municipal schools. These municipal schools follow the same state-approved educational programs and standards as do all of the municipal schools around the country. So the research results obtained from one school could be extrapolated to form an opinion on the situation in other schools as well. The authors worked only with one group from each school. Each of these groups attends programming classes which follow the traditional strategy of programming education and widely uses such environments as Turbo Pascal, Turbo Basic and QuickBasic for the educational process. Since the schools are municipal they use the same handbooks and manuals in their educational process and assign their students the same programming problems and tasks. Details as to the number of students in the groups are given in the Table 1.

First of all, the authors obtained statistical data concerning the average attendance of programming classes for the last five months prior to the beginning of this research. The purpose of these data was the exact evaluation of the students' current attitude towards the programming discipline. The statistical data obtained is presented in the Table 2.

Table 1. The number of students in the groups

Group No	Number of students
1	22
2	21
3	15
4	30
5	25

As is clearly seen from the Table 2, the attendance and therefore, the popularity of these programming classes, based on traditional programming teaching strategies, showed a permanent tendency for gradual decrease. In personal interviews, students named several factors behind their decision to stay away from the lessons. These factors coincided with those discovered by the authors themselves as having a negative impact on school programming education.

The next phase was to test the hypothesis H1. Teachers were asked to use Web-based programming environments in their classes instead of traditional desktop-based. Students were also allowed to use their smart phones to access the Web-based IDE. After two months the authors obtained new statistical data concerning the attendance of programming classes. The obtained data are given in the Table 3.

The Table 3 shows that the attendance of the classes increased to its normal level after they switched to using Web-based programming environments in their work.

The hypothesis H2 was tested at the extracurricular programming lessons held weekly in one of the schools. The aim of these lessons is to introduce students to the type of problems typically given at Programming Olympiads. Students can solve the problems using any programming language of their choice. Majority of these students prefer using JavaScript which is not taught and used at all in their regular programming classes.

Table 2. Attendance of the programming classes during five months

Group No	September	October	November	December	January
1	22	18	15	15	14
2	20	18	14	14	13
3	14	12	11	8	7
4	29	26	25	22	21
5	23	22	22	18	16

Table 3. Attendance of the classes after adopting Web-based IDEs

Group No	February	March
1	19	21
2	18	19
3	26	26
4	25	24
5	20	20

The hypothesis H3 was verified at the stage of testing the hypothesis H1. A significant number of students accessed Web-based programming environments using their smart phones. Possibility to accomplish classroom tasks within the environment of their favorite gadgets increases the attractiveness of programming classes and awakes students' interest.

The validness of the hypothesis H4 was tested by observing the students' process of solving the classroom problems. While working on typical problems of a mathematical nature students quickly lost their concentration and interest in completion of their task. On the contrary, when given simple chess problems they demonstrated the increased level of concentration and motivation and showed their interest in analyzing and evaluating the possible solutions.

A conclusion about the validness of the hypothesis H5 was made after reviewing the programs written at programming lessons. The authors made an observation that even the most accomplished students tend to use only a very limited subset of a programming language and this subset includes at most 20 percent of the overall functionality of the language. They find it irrelevant to study those features that lie beyond the range of their subset and are never used in their programming tasks.

4. Conclusion

Existing programming teaching strategies in schools are largely based on traditional approaches to the process of instruction. Many schools still use the same programming languages and environments as five or even ten years ago. Such stagnation in the development of a school-level programming education has a negative impact on the students' attitude towards programming and, therefore, their overall academic performance. Moreover, a negative experience obtained from programming lessons at school, has a potential to affect in the most negative way a student's attitude towards software engineering-related disciplines studied at the university level.

The main motivational factor behind this research was an attempt to propose effective methods which can be employed by a school teacher to increase the effectiveness and attractiveness of his programming classes. But, first of all, programming teachers should get acquainted with the six negative factors presented in this article which prevent their classes from bringing an educational value and awaking students' interests. Without elimination of these factors it is impossible to achieve the progress in the field of programming education. In each separate case the distribution of the influence of each separate factor on educational process can be varying so teacher should start with eliminating those factors which, in his opinion, have the biggest negative effect.

After eliminating these negative factors, the next phase for school programming teachers is to start employing new teaching methods and tools as suggested by the five hypotheses from the methods chapter. The correctness of these hypotheses were tested at programming lessons in local municipal schools and proved by statistical data obtained by the authors during the research. Ideally the most significant progress in the academic performance of students in a programming class can be achieved only when teachers strictly follow the recommendations contained in all of these hypotheses. But in most cases it is enough to try to follow the recommendations from the hypotheses H1 and H2 since the authors find them to be the most fundamental in the process of improving the programming teaching strategy.

Advancements in the field of teaching programming can bring educational benefits not only to the disciplines from computer-science curriculum, but to other school disciplines as well. If a student demonstrates a high level of motivation and interest at his programming lessons it means that he will try to solve as many different types of programming problems as possible within the limits of a school class. Doing this he will gradually increase his logical, cognitive and creative abilities and as a result get as close as possible to reaching his full personal and academic potential.

5. References

1. Purcell K, Heaps A, Buchanan J, Friedrich L. How teachers are using technology at home and in their classrooms. PewResearch Center. 2013 Feb. 2016. Available from: http://www.pewinternet.org/files/old-media/Files/Reports/2013/PIP_TeachersandTechnologywithmethodology_PDF.pdf
2. Flanagan D. JavaScript: The definitive guide. O'Reilly Media, 2001. 2016. Available from: http://kharchuk.ru/JavaScript.pdf.
3. Young Children, Apps and iPad. Michael Cohen Group LLC. 2016. Available from: http://sociallyspeakingllc.com/my-mission-for-socially/free-pdfs/a_study_of_young_children.pdf
4. Singh M. M-Learning: A new approach to learn better. International Journal of Education and Allied Sciences. 2010; 2(2):65–72.

5. Elias T. Universal instructional design principles for mobile learning. International Review of Research in Open and Distance Learning. 2011; 12(2):143–56. 2016. Available from: http://www.irrodl.org/index.php/irrodl/article/view/965/1675
6. Sakibayev SR, Sakibayeva BR, Toibazarov DB. Android-based mobile device as a programming environment in a school programming class. Indian Journal of Science and Technology. 2016; 9(20). DOI: 10.17485/ijst/2016/v9i20/94488.
7. Savill-Smith C, et al. Mobile learning in practice: Piloting a mobile learning teachers' tool kit in further education colleges. 2010. 2016. Available from: https://edugalaxy.intel.ru/index.php?act=attach&type=blogentry&id=1437
8. Saylor M. The mobile wave: How mobile intelligence will change everything. Perseus Books. Vanguard Press. 2012, 176. 2016. Available from: https://www.amazon.com/Mobile-Wave-Intelligence-Change-Everything/dp/0306822539
9. Crescente ML, Lee D. Critical issues of M-Learning: Design models, adoption processes and future trends. Journal of the Chinese Institute of Industrial Engineers. 2011; 28(2):111–23.
10. Sharma SK, Kitchens FL. Web services architecture for M-Learning. Electronic Journal on E-Learning. 2004; 2(1):203–16. 2016. Available from: http://ubimotion.iwi.uni-hannover.de/lv/seminar_ws05_06/files/10_Riemer/literatur/sharma-kitchens2004.pdf
11. Freeman Ch. Nim: Serious math with a simple game. Prufrock Press Inc; 2005.
12. Schildt H. Turbo C: The complete reference. Osborne: McGraw-Hill; 1998. 2016. Available from: http://home.iitj.ac.in/~ug201310035/H.Schildt.Cpp.The.Complete.Reference.4th.Edition.ENG.pdf .
13. McFarland DS. JavaScript and JQuery: The missing manual. O'Reilly Media. 2014. 2016. Available from: https://books.google.com.ua/books?id=0oWMBAAAQBAJ&printsec=frontcover&hl=ru#v=onepage&q&f=false