Mobile technologies and programming in terms of the continuity principle in school and university education

Eleonora Brtka[1], Igor Vecštejn[2], Vladimir Brtka[3], Vesna Makitan[4] and Ivana Berković[5]

Abstract – **The paper is dedicated to the issues of teaching mobile application development and, as a consequence, training of highly qualified mobile developers. Nowadays, training professional mobile developers is a crucial task all over the world. The researchers emphasize the complexity of mobile application development associated with its multidisciplinary, the mobile device hardware limitations, the necessity of object-oriented programming in the mobile development.**

Keywords – **Mobile application development, object-oriented programming, educational technology, modern teaching.**

I. Introduction

Mobile computing is one of the rapidly evolving fields of computer science [1]. Teaching and learning the development of mobile applications is one of the recent problems [2]. The demand for competent mobile application developers globally is high. Many researchers agree that training mobile application developers at the secondary level is impossible for several reasons [3]. First, mobile application programming was a multidisciplinary field that encompassed knowledge of areas such as software development, human-computer interaction, web programming, IT security, network interaction, artificial intelligence, machine learning. Second, the professional development of mobile applications requires knowledge and skills of object-oriented programming [3, 4]. The insufficient level of programming background makes necessary to bridging gap in students' knowledge of the object oriented programming basic concepts within the initial undergraduate courses. It also leads to teaching about the development of mobile applications in higher education courses. As a result, students are limited in time to professionally master the development of mobile applications.

Due to the above circumstances, it is necessary to overhaul school and university curricula in terms of the principle of continuity and filling gaps in programming and to quickly adapt students to modern learning content in the development of mobile applications.

In this paper, we dealt with the question: How to bridge the gap in the level of high school graduates' knowledge on programming in order to prepare successful mobile application developers at the university? To answer such a question, we must analyze similar papers that dealt with the problem of the continuity of the knowledge gap of high school students, quickly adapted to university disciplines and effectively developed the professional competencies of future professionals.

II. Related work

Many studies [5, 6, 7, 8] show the necessity, possibility and popularity of studying programming in high school. The teaching of programming in high school is justified, because in this age, according to Piaget, there is a transition between phases from concrete thought operations to abstract logical thinking [8].

Therefore, learning programming can begin at this age, but with the right content, in order to gradually learn from simple to complex. At the same time, when developing curricula, it should be taken into account that in the teaching of computer science in school, the main goal should be teaching the basic concepts transmitted by the language, and not teaching the language itself [7]. At the same time, new strategies are needed in teaching programming, because learning computer programming is not easy, even for students who have enrolled in computer science disciplines. Students who have a basic knowledge of programming find advanced programming courses difficult, because these courses require higher-order thinking skills [9]. In such a case, it is considered interesting to approach the teaching of programming based on the use of software for visualization of algorithms such as Scratch, Alice, App Inventor. These platforms facilitate programming learning, making the programming process engaging and visualizing enabling the development of knowledge about basic programming principles, basic concepts of object-oriented programming, and basic principles of mobile application programming [9]. After learning these platforms, students can safely move not only to more complex programming languages but also to complex processes of mobile application development [10].

A visual programming environment, such as Scratch, is designed to teach students ages 8 and up. Scratch motivates learning of programming and enables mastering of basic principles of programming [10, 11]. Scratch enables the creation of animated and interactive applications without writing program code, revealing the creativity of school children and highly motivating them to learn programming

[1]Eleonora Brtka is with University of Novi Sad, Technical faculty "Mihajlo Pupin", Đure Đakovića bb, 23000 Zrenjanin, Serbia, E-mail: eleonorabrtka@gmail.com

[2]Igor Vecštejn is with University of Novi Sad, Technical faculty "Mihajlo Pupin", Đure Đakovića bb, 23000 Zrenjanin, Serbia, E-mail: igor.vecstejn@gmail.com

[3]Vladimir Brtka is with University of Novi Sad, Technical faculty "Mihajlo Pupin", Đure Đakovića bb, 23000 Zrenjanin, Serbia, E-mail: vbrtka@tfzr.uns.ac.rs

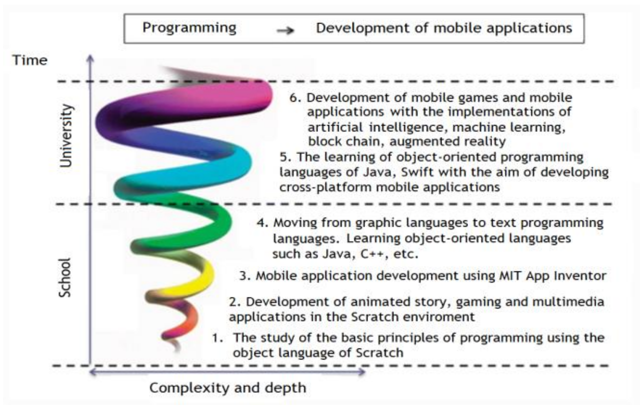
[4]Vesna Makitan is with University of Novi Sad, Technical faculty "Mihajlo Pupin", Đure Đakovića bb, 23000 Zrenjanin, Serbia, E-mail: vesna@tfzr.uns.ac.rs

[5]Ivana Berković is with University of Novi Sad, Technical faculty "Mihajlo Pupin", Đure Đakovića bb, 23000 Zrenjanin, Serbia, E-mail: ivana.berkovic62@gmail.com

[11]. The results of the research indicate the possibility and need to use this environment in school for the development of programming skills, algorithmic and logical thinking [11, 12]. In addition, K-12 schools around the world, and even some universities (including Harvard and the University of California, Berkeley), use Scratch as a first step in programming [13]. To enable a smooth transition from programming in visual environments (Scratch, Alice) to creating Android applications, MIT App Inventor can be used.

App Inventor (AI) is a visual programming environment that allows users to easily develop mobile apps for Android-based smartphones without writing code. This environment can be studied in an information technology course, because AI is easy to learn, accessible and helps students solve problems [14]. The advantage of using such a tool is the great motivation of students to develop mobile applications. In addition, students acquire knowledge about the development of interface design, object programming and events in the development of mobile applications [15].

Taking into account the opinions of researchers on the need to bridge the knowledge gap through the principle of continuity in learning content [16, 17], motivation to learn programs in primary and secondary schools [5, 6], the effectiveness of learning programs in Scratch and further transition to development of mobile applications in App Inventor [9, 10, 13], this paper proposes a spiral model of teaching programming and development of mobile applications in high school and university in the context of the Serbian education system, which can be used in countries with similar education systems [18]. A spiral model of teaching mobile application development is presented in Fig 1.



Programming → Development of mobile applications
Time
University
School
6. Development of mobile games and mobile applications with the implementations of artificial intelligence, machine learning, block chain, augmented reality
5. The learning of object-oriented programming languages of Java, Swift with the aim of developing cross-platform mobile applications
4. Moving from graphic languages to text programming languages. Learning object-oriented languages such as Java, C++, etc.
3. Mobile application development using MIT App Inventor
2. Development of animated story, gaming and multimedia applications in the Scratch enviroment
1. The study of the basic principles of programming using the object language of Scratch
Complexity and depth

Fig. 1. Spiral model of teaching mobile application development

III. Computer science content and programming learning

Currently, Serbian schools are moving to updated contents of curricula, in the context of integration into world educational practice, while maintaining the best traditions and standards of national education. According to the updated curriculum, the compulsory subject of informatics and computer science was introduced in primary school, and the subject of informatics was innovated in secondary schools. The subject of informatics and computer science is a practical discipline that emphasized the development of skills for efficient and correct use of information and communication technology tools (mobile phones, computers, players, digital cameras, video cameras, etc.) in learning activities and everyday life. Informatics in high schools is a theoretical discipline that focuses on methods and processes of information transformation using computers. The curriculum of informatics has been developed on the basis of the spiral principle, according to which most of the goals and topics of learning after certain academic periods of teaching (during the school year or in the following classes) are repeated at increasingly complex levels.

Programming is taught from the 5th grade of elementary school to the 4th grade of high school. Content includes learning:

1. game programming environments - Scratch (grades 5-6);
2. integrated environment for software development and high-level programming languages C / C ++, Python, Delphi, Lazarus, etc. (7th grade of primary school and 1st grade of secondary school);
3. object oriented programming with C / C ++, Python, Delphi, Lazarus, etc .; web programming with HTML, XML, scripting language (2nd to 4th grade of high school); development of mobile applications (3rd to 4th grade of high school).

In addition, analysis of the goals of high school curriculum on computer science shows several shortcomings of the updated curriculum on programming such as:

1. The choice of programming environment is made without proper thinking. Thus, from 5th to 6th grade, it is suggested to learn visual programming environments such as Scratch, which enable the programming of multimedia stories, games, without writing code;
2. Emphasis is placed only on mastering the programming of linear, branched and loop structures, classification of data types, one-dimensional arrays, components of the integrated development environment;
3. In 3rd and 4th grade of high school, the topic of “Mobile Application Development” is covered, but no mobile application development environment is considered and only a few hours are allocated.

Updated computer science curricula in high school offer more modern content, focused on world educational paths and correspond to state-of-the-art computer technology. However, it is necessary to contribute to creating a seamless continuum between educational levels and well-thought choose environments and topics on programming with a further focus on the development of smart devices and mobile applications.

IV. Results

During the summer IT school, experimental training was conducted on the course "Fundamentals of iOS application development" for first year students of the IT faculty. A total of 72 people were included. To conduct a comparative analysis, students were divided into three groups: students as teachers of computer science, students of IT majors and IT-faculty. Before training preliminary testing and survey were conducted to determine the level of basic knowledge of programming. Testing showed a very low level of basic knowledge of students - teachers of computer science - at the level of 48%. Students of IT-majors have a basic programming knowledge of 76%. IT faculty showed the highest result of 97%. In order to answer the research question, an analysis of knowledge was held in order to determine in which topics students have shortcomings and how this affects the achievements in the development of mobile applications. The preliminary assessment test included questions on basic topics, namely knowledge of algorithmic structures, software development life cycle phases, code debugging, input-output operators, loop operators, mathematical functions, principles and concepts of object-oriented programming.

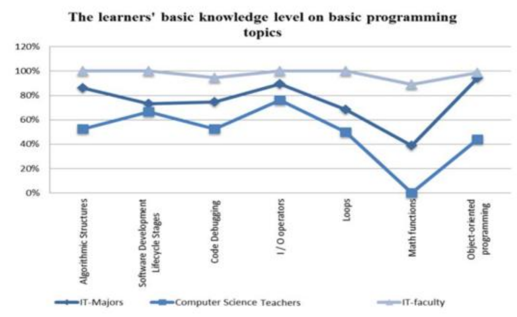


Fig. 2. The results of preliminary testing on programming in the context of basic topics

As shown in Fig. 2 Computer science teachers are at the lowest level of basic knowledge in all topics. They show an insufficient level of knowledge, less than 50%, on such topics as loop operators, mathematical functions, principles and concepts of object-oriented programming. On the topics of algorithmic structures, phases of the software development life cycle, elimination of program code errors, input-output operators show the average level of basic knowledge. Such a low level of basic knowledge is explained by the fact that high school students learned only the Pascal programming language for a short duration of the course (Fig. 3).

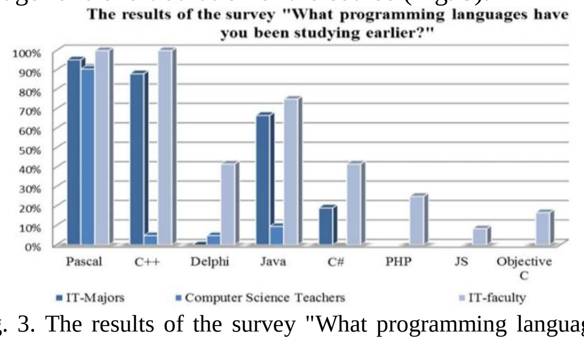


Fig. 3. The results of the survey "What programming languages have you been studying earlier?"

Students of IT-majors have an insufficient level of basic knowledge about "Mathematical functions" (less than 50%). In topics such as Software Development Lifecycle Stages, Code Troubleshooting, Loop Operators they display an average level of knowledge. In the topics of Algorithmic Structure, Principles and Concepts of Object Oriented Programming have a high level of basic knowledge.

This Gap in the knowledge of IT- majors and computer science teachers is a consequence of the Gap in curricula. At school all students learned Pascal, and in the first year IT students, unlike teachers, learned Java, C #, C ++, Delphi (Fig. 3).

The highest level of knowledge is displayed by faculty who have a certain experience in teaching programming languages and software development. In all topics, the level of basic knowledge is sufficient and high (Fig. 2). As shown in the diagram (Fig. 3), the faculty pointed to the knowledge of the languages Pascal, C ++, C#, Java, Delphi, PHP, Objective C.

Thus, preliminary testing showed different levels of basic programming knowledge, depending on the level of high school and university training. To determine the impact of the basic programming knowledge on the further achievement of mobile application development, it is need to consider the content and learning outcomes of the course Basics of iOS apps development.

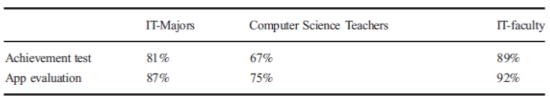
During the practical course "Basics of iOS application development", summer school participants learned how to develop mobile applications in the X-code environment in the Objective-C language. The following topics were covered during the course to lay the foundation for knowledge about mobile application development:

1. Basics of creating a Single View application
2. UI controllers
3. Use of multimedia
4. Sensors in the application
5. Programming complex View
6. SQLite database
7. Final project

During the course, participants learned how to develop mathematical models and algorithms for mobile applications using demo examples of mobile applications. During the practical session, students created applications in accordance with the instructions, tested them and eliminated errors. To consolidate skills within individual work, students developed their own applications, working in a small team. Such organization of the course enabled the implementation of a spiral model, which proposes the enrichment and improvement of developed mobile applications.

The knowledge and skills to develop mobile applications was assessed in two ways: achievement test and criterial assessment of developed projects. Assessment of mobile applications took into account the following criteria: understanding of the task, the correctness of the algorithm for problem solving, application logic, programing technique, user interface design style, teamwork, self sufficiency of teamwork. The results of achievement test and evaluation developed projects for the three groups are shown in Table 1.

TABLE I
THE RESULTS OF THE ACHIEVEMENT TESTING AND EVALUATION APPLICATION OF COURSE PARTICIPANTS (AVERAGE VALUE)



	IT-Majors	Computer Science Teachers	IT-faculty
Achievement test	81%	67%	89%
App evaluation	87%	75%	92%

As can be seen from the table, computer science teacher are still lagging behind in the knowledge of mobile application development from IT-Majors. Observation and achievement test results showed that students found difficulty in developing application logic, coding and debugging a mobile application. This indicates a poor development of CT among students, that is, the ability to analyze and solve a problem, logically analyze the program code and gain experience from it. However, a special approach to the course teaching, based on the spiral principle and collaborative interaction between lecturer and students, allowed to achieve the learning objective and to motivate computer science teachers. The survey results show a high motivation (100%) to continue learning the course mobile development, which is explained by the popularity of the mobile computing field.

V. CONCLUSION

Studying international experiences in overcoming the knowledge gap of students at high school and university level, we can conclude that one of the possible approaches to solving the research issue is rearranging school and university curricula with progressive improvement and development of knowledge from basic programming for mobile application development. Thanks to this approach, teaching units that are established at the basic level are progressively improved from one educational phase to another and develop appropriate skills. In this regard, it is advisable to implement a spiral model of teaching the development of mobile applications, according to which the content of learning in high school and university should be planned in accordance with the spiral principle [19]. This model should be applied to all IT students, including computer science teachers who must master mobile technologies at a professional level in order to be able to apply them in the educational process.

ACKNOWLEDGEMENT

Ministry of Education, Science and Technological Development, Republic of Serbia financially supported this research, under the project number TR32044: “The development of software tools for business process analysis and improvement”, 2011-2020.

REFERENCES

[1] Pinar, Muyan-Özçelik (2017). A hands-on cross-platform mobile programming approach to teaching OOP concepts and design patterns, Proceedings of the 1st International Workshop on Software Engineering Curricula for Millennials, May 20–28, Buenos Aires, Argentina.

[2] Z. Nurbekova, G. Aimicheva, (2018) «Teaching Mobile Application Development: from the Idea to the Result», 3rd International Conference on Computer Science and Engineering (UBMK) IEEE, pp. 666-669, 2018.

[3] Taft, D. K. (2007). Programming grads meet a skills gap in the real world.

[4] Alston, P. (2012). Teaching Mobile Web Application Development: Challenges Faced And Lessons Learned, In: Proceedings of 13th Annual Conference on Information Technology Education (SIGITE ‘12), 239–244.

[5] Dillashaw, F., & Bell, S. (1985). Learning outcomes of computer programming instruction for middle-grades students: A pilot study, Proceedings of the 58th annual meeting of the National Association for research in science technology.

[6] Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2018). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. Journal of Educational Psychology.

[7] Kafai, Y., & Burke, Q. (2013). Computer programming goes back to school. Phi Delta Kappan, 95(1), 61–65.

[8] Inhelder, B., & Piaget, J. (1958). An essay on the construction of formal operational structures. The growth of logical thinking: From childhood to adolescence (A. Parsons & S. Milgram, Trans.). New York: Basic Books.

[9] Saltan, F. (2016). Looking at algorithm visualization through the eyes of pre-service ICT teachers. Universal Journal of Educational Research, 4(2), 403–408.

[10] Cheung, J., Ngai, G., Chan, S., and Lau, W. (2009). Filling the gap in programming instruction: A text-enhanced graphical programming environment for junior high students, SIGCSE Symposium on Computer Science Education, Chattanooga, TN, March 2009, pp. 276–280.

[11] Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. ACM Transactions on Computing Education (TOCE), 10(4), 16.

[12] Funke, A., Geldreich, K., & Hubwieser, P. (2017). Analysis of scratch projects of an introductory programming course for primary school students. In 2017 IEEE Global Engineering Education Conference (EDUCON) (pp. 1229-1236). IEEE.

[13] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . & Kafai, Y. B. (2009). Scratch: Programming for all. Communications of the ACM, 52(11), 60–67.

[14] Morelli, R., de Lanerolle, T., Lake, P., Limardo, N., Tamotsu, E., & Uche, C. (2011). Can android app inventor bring CT to k-12. In Proc. 42nd ACM technical symposium on Computer science education (SIGCSE'11).

[15] Wagner, A., Gray, J., Corley, J., and Wolber, D., (2013). Using App Inventor in a K - 12 Summer Camp, SIGCSE '13 , 621–626.

[16] Stone, J. A. (2019). Student perceptions of computing and computing majors. Journal of Computing Sciences in Colleges, 34(3), 22–30.

[17] Dodero, J.M., Mota, J.M., & Ruiz-Rube, I. (2017). Bringing CT to teachers' training: A workshop review. In Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality (p. 4). ACM.

[18] Aimicheva, G., Kopeyev, Z., Ordabayeva, Z. et al. A spiral model teaching mobile application development in terms of the continuity principle in school and university education. Educ Inf Technol 25, 1875–1889 (2020).

[19] Bruner, J. S. (1960). The process of education. Cambridge: Harvard University Press.