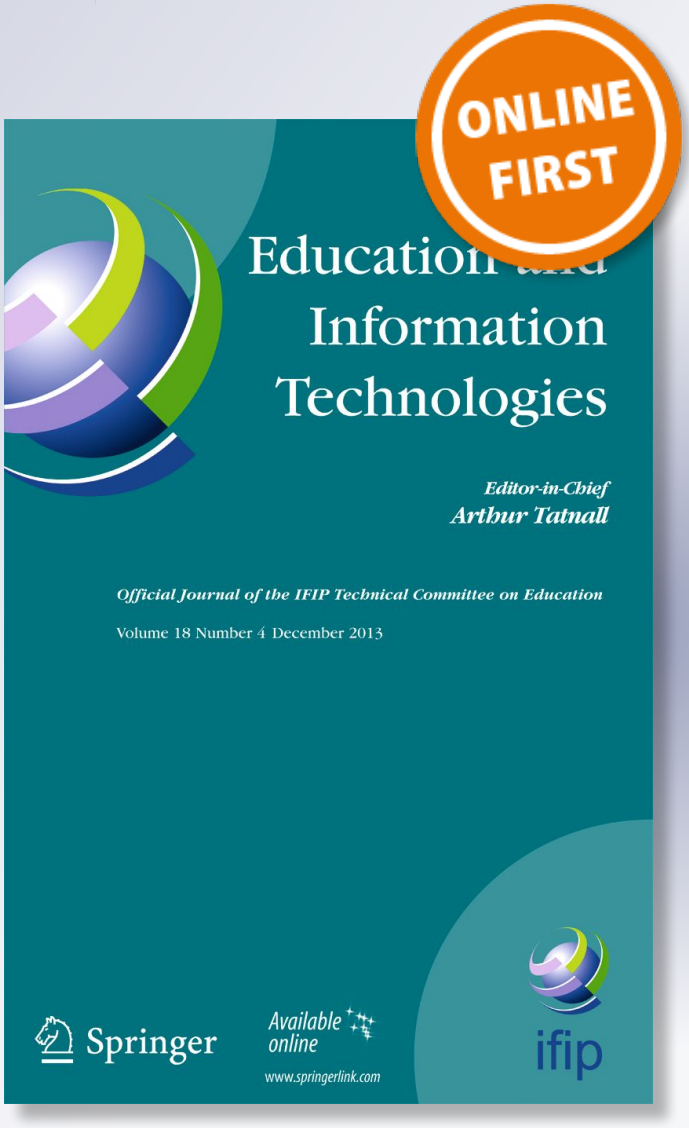
A spiral model teaching mobile application development in terms of the continuity principle in school and university education

G. Aimicheva, Zh. Kopeyev, Zh. Ordabayeva, N. Tokzhigitova & S. Akimova

Education and Information Technologies
The Official Journal of the IFIP Technical Committee on Education

ISSN 1360-2357

Educ Inf Technol
DOI 10.1007/s10639-019-10051-z



Springer

Education and Information Technologies
https://doi.org/10.1007/s10639-019-10051-z

A spiral model teaching mobile application development in terms of the continuity principle in school and university education

G. Aimicheva[1] · Zh. Kopeyev[1] · Zh. Ordabayeva[2] · N. Tokzhigitova[3] · S. Akimova[4]

Received: 7 August 2019 / Accepted: 30 October 2019/Published online: 27 November 2019

Abstract
The article is devoted to the issues of teaching mobile application development and, as a consequence, training of highly qualified in-demand 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 multidisciplinarity, the mobile device hardware limitations, the necessity of object-oriented programming in the mobile development. Due to the complexity of the mobile development field and the gap in programming knowledge of first-year students, there are fears that prepare highly qualified mobile developers during undergraduate education is impossible. In this regard, the article proposes a spiral model teaching mobile application development with the aim of effective training of mobile developers. The spiral model covers all levels of teaching programming from high school to higher education with aim to develop knowledge from introductory programming to mobile application development. The offered spiral model suggests the continuity in the content and overcoming the gap in programming knowledge between high school and higher education. Such a model is the most appropriate for the training of highly qualified mobile developers in the context of Kazakhstan's education system.

Keywords Mobile application development (MAD) · GAP in programming knowledge · Continuity in teaching MAD · Mobile developers · Spiral model

1 Introduction

Mobile computing is one of the rapidly developing areas of computing (Pinar 2017). Teaching and learning the mobile application development is one of the recent

✉ G. Aimicheva
aimicheva@mail.ru

Extended author information available on the last page of the article

problems (Nurbekova and Aimicheva 2018). The demand for competent mobile application developers globally is high. Many researchers agree that the training of mobile application developers under the undergraduate programme is impossible for several reasons (Taft 2007). First, mobile application development is a multidisciplinary field that integrates knowledge of such areas as software development, human-machine interaction, web programming, IT-security, network interaction, artificial intelligence, machine learning. This reason provides to study mobile application development only in undergraduate courses. Secondly, the professional mobile application development requires the knowledge and skills of object-oriented analysis and programming (Pinar 2017; Alston 2012). 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 mobile application development in upper division courses. As a result, students are time poor to master the mobile application development professionally.

Due to the aforementioned circumstances, the overhaul of school and university curricula is required in terms of the continuity principle and filling gap in programming and to quickly adapt students to contemporary learning content of mobile application development.

The following research question will be addressed in this study: 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?

Answer such a question we need to analyze similar works that addressed the issue of continuity of knowledge in various disciplines and educational levels in order to bridge the knowledge gap of high school graduates', quickly adapt to university disciplines, and effectively develop the professional competencies of future specialists.

In the work (Stone 2019). it is noted that in the last decade, due to an increased recognition of the importance of computing skills in the global economy, there has been a triple increase in the undergraduate Computer Science (CS) program enrollments by three times more. Most of the incoming students have an incorrect perception of computing, computing majors, and computing-related careers due to insufficient career counseling and applications-centric high school computing curricula. Survey has shown that 13.04% of participants taking a computer programming course in high school. This fact directly reflects on the students' achievement on computer science in university. The authors see the solution to the problem in curriculum overhaul in order to develop more in-depth knowledge of computer science in the school course and to successfully adapt to university content on computing majors and in accordance with the growing requirements of global economics.

Many studies confirm the need for learning programming as a way of developing computational thinking (CT) (Buitrago Flórez et al. 2017; Dodero et al. 2017; Fronza et al. 2019; Bers 2017; Good et al. 2017; Shute et al. 2017). CT was first used by Seymour Papert (Papert 1980; Papert 1991) and was introduced as a term by Wing (2006). It is known that CT is the important skill of modern employee, not only in computing majors, but also in all other fields. So, the authors of works (Buitrago Flórez et al. 2017; Dodero et al. 2017; Bers 2017; Shute et al. 2017) established the impact of learning programming on the developing of CT at school age pupil. In this basis they state that programming be taught from elementary school to universities in order to develop soft-skills, that future professional employee should possess, such as problem

solving, logical, algorithmic and critical thinking. In this case, programming used as a vehicle for develop CT, and the aim of researchers is to choose an educational strategy, that provided progressive enhancement of CT skills from the school level to the university level, combining different environments and programming languages.

The work (Zorana 2003) is considered a method for solving the problem of gap in physics knowledge for engineering students. In order to bridge the knowledge gap in physics in higher school and further effective study at university, the author considers to develop information literacy through the collaborative work of teachers, lecturers and librarians and to introduce changes in curricula at different educational levels. This paper emphasizes the role of teacher collaboration at different educational levels and continuity of learning content.

In the work (Siadaty and Taghiyareh 2007), the author states that teaching students with different domain knowledge, learning styles, interests and preferences is impossible using the «one-size-fits-all» approach. The use of an adaptive e-learning system is proposed to bridging the gaps in the students' knowledge and to improve the learning outcomes. This system offers students pedagogically based individual learning content due to the basic knowledge of the subject area and student's learning style. It has been experimentally proven that learning content, adapted to the needs of students, improved learning outcomes compared to using common learning content for all students.

The authors (Offir et al. 2003) propose the use of distance learning for high school students of peripheral areas to provide their high learning potential in the computer science and adapt to learning at university. As part of the project, high school students were invited to study at the initial computer science course at the university. Educational content includes audio and video materials that are available for study and download on the project website, as well as tasks based on active online performance to enhance the cognitive abilities of high school students. The online collection and analysis of the results of the assignments allowed the university faculty, together with the school teachers, to adjust the learning process and adapt high school students to university studies. The authors of the paper (Offir et al. 2003) emphasize the importance of collaboration of school teachers and university faculty and the adaptation of learning content to bridge the gap in students' knowledge and achieve better learning results.

Having studied international experience in bridging the gap in students' knowledge at high school and university, we can conclude that one of the possible approaches to solve the research question is to overhaul school and university curricula with a progressive enhancement and develop knowledge. In other words, it is necessary to provide for the continuity of the learning content from school to university level. In this case, the most acceptable approach to the development of the learning trajectory of programming teaching and the successful training of mobile application developers is the spiral approach. Due to this approach, the learning unit of knowledge establish at the elementary level, is progressive enhanced from one educational level to another and developing appropriate skills, as a consequence, curricula should be developed in terms of the spiral principle (Bruner 1960).

The spiral approach was suggested by psychologist Jerome Bruner (1960), who considers that: «Any subject can be taught effectively, in some intellectually honest way, to any child at any stage of development» (Bruner 1960). The approach effectiveness is due to the fact that it involves multiple revisions of topics with increasing depth and complexity (Araujo et al. 2018), thereby allowing basic knowledge to be

deposited in a long-term memory (Joshi and Desai 2016). In addition, the spiral approach assumes horizontal and vertical integration of knowledge, which allows using the previous knowledge of students on certain topics at each stage and reducing cognitive overload due to the separation of content depending on its complexity (Araujo et al. 2018; Coelho & Moles et al. 2016). Moreover, researchers consider that the spiral learning model is an interactive method of teaching supports learning by making students to build on what is already familiar to them rather than learning new and difficult things (Joshi and Desai 2016).

In order to realize the spiral model of programming teaching at school and the successful training of mobile application developers at the university, it is necessary to single out the knowledge that is basic for the development of mobile applications, this is knowledge of the basic principles of programming, basic concepts of object-oriented programming and the fundamentals of mobile application development. This basic knowledge should be laid as a part of a school course in computer science and further deepening at the university. Therefore, it is necessary to revise the content of curricula in computer science of secondary education and study programs of higher education of the Republic of Kazakhstan in order to build a more effective educational strategy for the training of highly qualified specialists.

2 The learning programming: A literature review

Many studies (Dillashaw and Bell 1985; Scherer et al. 2018; Kafai and Burke 2013; Inhelder and Piaget 1958; Yardi and Bruckman 2007; Tan et al. 2009) show the necessity, possibility and popularity of studying programming in high school. Teaching programming in middle school is justified, because at this age, according to Piaget's theory, there is a transition between stages from concrete thought operations to abstract logical thinking (Inhelder and Piaget 1958). Therefore, teaching programming can be started at this age, but with the right content, in order to gradually learn from the simple to the complex in terms of scaffolding principle. At the same time, when developing programming curricula, one should take into consideration that in teaching computer science at school the main goal should be teaching fundamental concepts conveyed by the language, and not teaching the language itself (Kafai and Burke 2013). At the same time, new strategies are needed in teaching programming, because the learning of computer programming is not easy, even for university students enrolled in computer-related disciplines. Students, who have basic knowledge of programming, perceive in-depth programming courses as difficult, because these courses require higher-order thinking skills (Saltan 2016).

In such a case, an approach to teaching programming based on the use of algorithm visualization software such as Scratch, Greenfoot, Alice, App Inventor is considered to be interesting. These platforms facilitate the learning programming, making the programming process engaging and visualizing and allow developing knowledge of the basic programming principles, the basic concepts of object programming and the basic principles of programming mobile applications (Saltan 2016). After learning these platforms, students can safely to go on not only to more complex programming languages, but also to complex processes of the mobile application development (Cheung et al. 2009).

The visual programming environment, such as Scratch, is designed to teach programming to students ranging from 8 years old or above. Scratch motivates to learning programming and provides to master the basic principles of programming (Cheung et al. 2009; Maloney et al. 2010). Scratch allows to create animated and interactive applications without writing program code, revealing creativity of schoolchildren and highly motivating them to learn programming. (Maloney et al. 2010). The research results indicate the possibility and necessity of using this environment in school to develop programming skills, algorithmic and logical thinking (Maloney et al. 2010; Funke et al. 2017). In addition, K-12 schools around the world and even some universities (including Harvard and the University of California at Berkeley) use Scratch as the first step into programming (Resnick et al. 2009). To provide smooth transition from programming in visual environments (Scratch, Greenfoot, Alice) to the creation of Android applications the MIT App Inventor can used.

App Inventor (AI) is a visual programming environment that allows users to easily develop mobile applications for Android-based smart phones without writing any program code. This environment can be studied in a computer science course, because AI is easy to learn, accessible and helps students solve problems, rather than coding (Maloney et al. 2010; Morelli et al. 2011). The benefit of the using such tool is in the high motivation of students to develop mobile applications. In addition, students acquire knowledge of interface design development, object-event programming in mobile application development (Wagner et al. 2013). In the work (Karakozov and Manyakhina 2016), an analysis of South Korean textbooks on computer science of primary and secondary schools is carried out. The textbooks provide content on teaching schoolchildren to program and develop mobile applications using Scratch and App Inventor. Middle school students (Grades 5–7) having gained in elementary school experience in developing animated stories and games in the Scratch environment, move on to developing applications for mobile devices and programming machines and robots. This is a non-standard approach that offers students learning content according to modern information technologies. It is much more interesting for students to develop applications for mobile devices that they use in their daily life than for a personal computer. In addition, the principles of work in the App Inventor are similar to Scratch, therefore, it provides an easy transition to it. The authors of the textbook do not set a goal to teach students to develop complex applications for mobile devices; they aim at popularly explain the basics of programming mobile applications. During the course, the students acquire wide knowledge of the programming algorithms and applying them in building Android based applications. Possessing such a set of programming prerequisites, it will be easy for a student to adapt to the university learning content and to master more complex topics on mobile application development.

Considering opinions of researchers on necessity to bridging the knowledge gap through continuity principle in learning content (Stone 2019; Buitrago Flórez et al. 2017; Dodero et al. 2017; Zorana 2003; Siadaty and Taghiyareh 2007; Offir et al. 2003), motivation for learning programming in primary and secondary schools (Dillashaw and Bell 1985; Scherer et al. 2018; Kafai and Burke 2013; Inhelder and Piaget 1958; Yardi and Bruckman 2007), the efficiency of learning programming in Scratch and further move to the mobile application development in App Inventor

(Saltan 2016; Cheung et al. 2009; Maloney et al. 2010; Funke et al. 2017; Resnick et al. 2009; Morelli et al. 2011; Wagner et al. 2013; Karakozov and Manyakhina 2016) the paper proposes a spiral model teaching the programming and mobile application development in high school and university in context of the education system of Kazakhstan, which can be used in countries with similar education system (see Fig. 4).

In the next section we will consider the content of the current computer science curricula and the ensuing challenges.

3 Computer science content and programming training in Kazakhstan

At present, Kazakhstani schools pass to updated curricula content, in context of integration into the world educational space and compliance the national education system with the standards of world educational practice, while maintaining the best traditions and standards of national education. According to update curriculum the Information and communications technology (ICT) course has been introduced in primary school and the computer science course in secondary and high schools (Standard curriculum on the subject, n.d.-a, Standard curriculum on the subject, n.d.-b). The «ICT» course is a practical discipline that emphasized on the development of skills for the effective and correct using of information and communication technology tools (mobile phones, computers, players, digital cameras, video cameras, etc.) in the learning activities and everyday life and the «Computer science» is a theoretical discipline that focuses on the methods and processes of transforming information using computers.

The computer science curriculum is developed on the basis of the spiral principle, according to which most of the learning objectives and topics after certain academic periods of teaching (during the school year or in the following classes) are repeated again at increasingly complex levels.

Programming falls under the head of «CT» which is learned from grades 5 to 11. The content includes learning the following such as:

1. ***game programming*** environment Logo, Scratch (Grades 5–6);
2. ***integrated development environment*** for software development and high level programming languages C/C++, Python, Delphi, Lazarus etc. (Grades 7–9);
3. ***object-oriented programming*** with C/C++, Python, Delphi, Lazarus, etc.; ***web programming*** with HTML, XML, script language (grades 10–11); ***mobile application development*** (grades 10–11).

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 due reflection. So, in grades 5–6, it is proposed to learn the visual programming environments such as Logo, Scratch, which allow programming multimedia stories, games, without writing the code. In Scratch students programming with visual blocks and further move to integrated development environments will be difficult, since it requires coding skills in high-level C / C ++ languages, Python, Delphi, Lazarus, and higher-order thinking skills (Saltan 2016);

2. The emphasis is made only on mastering the programming of a linear, branching and loop structures, the classification of data types, the one-dimensional arrays, components of an integrated development environment;
3. In Grades 10–11, the theme «Mobile Application Development» is covered, but no one environment for mobile application development is considered and only few hours are allocated.

The updated computer science curricula in high school offers more modern content, focused on world educational trajectories and suit state-of-art computing 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.

4 Results and discussion

During the summer IT school, experimental training was conducted on the course «Basics of iOS apps development» for the first-year students and IT-faculty. A total of 72 people were covered. To conduct a comparative analysis, the students were divided into three groups: students as Computer science preservice teachers, 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 - Computer science preservice teachers - at the level of 48%. Students of IT-majors have a basic programming knowledge of 76%. IT-faculty showed the highest result of 97% (see Fig. 1).

To answer the research question, knowledge analysis was held to find out in which topics students have gaps and how this impacts on achievement on the mobile application development.

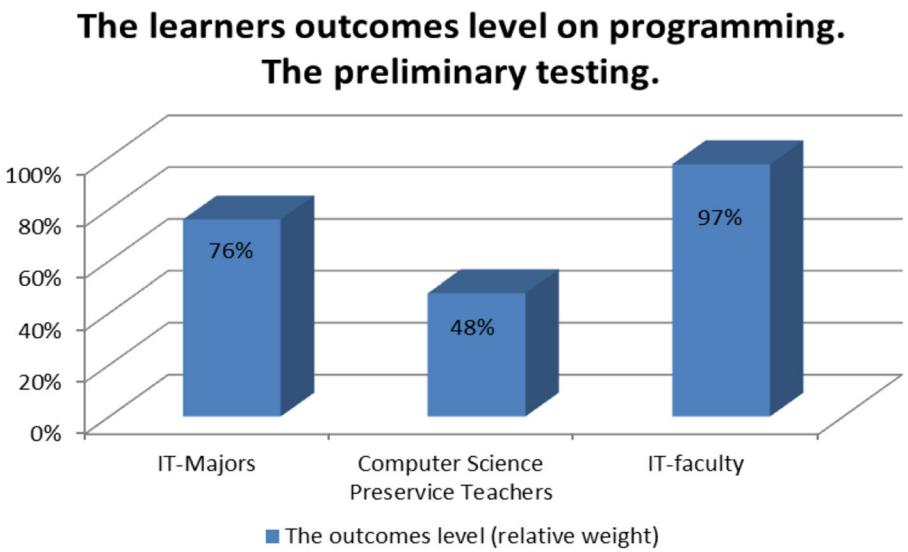


Fig. 1 Outcomes of preliminary testing of students for the level on programming

The preliminary assessment test contained questions on basic topics, namely knowledge of algorithmic structures, stages of the software development life cycle, code debugging, input-output operators, loop operators, mathematical functions, principles and concepts of object-oriented programming.

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

Students of IT-majors have an insufficient level of basic knowledge (less than 50%) on the «Mathematical functions». In such topics as «The stages of the software development life cycle», «Code debugging», «Loop operators» they display the average level of knowledge. In the topics of «Algorithmic structures», «Principles and concepts of object-oriented programming» they have a high level of basic knowledge.

This Gap in the knowledge of IT- majors and Computer science preservice teachers is a consequence of the Gap in curricula. At school all students learned Pascal, and in the first year IT students, unlike preservice 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.

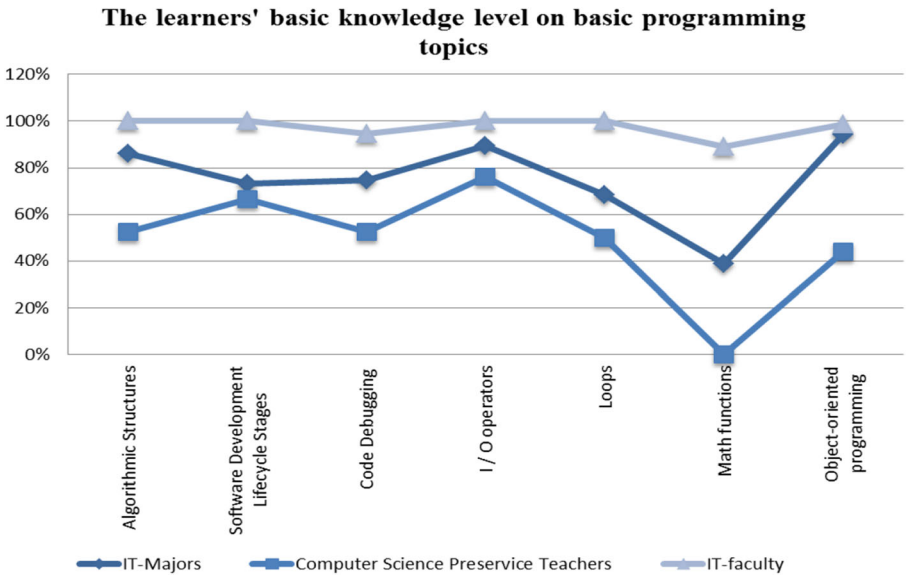


Fig. 2 The results of preliminary testing on programming in the context of basic topics

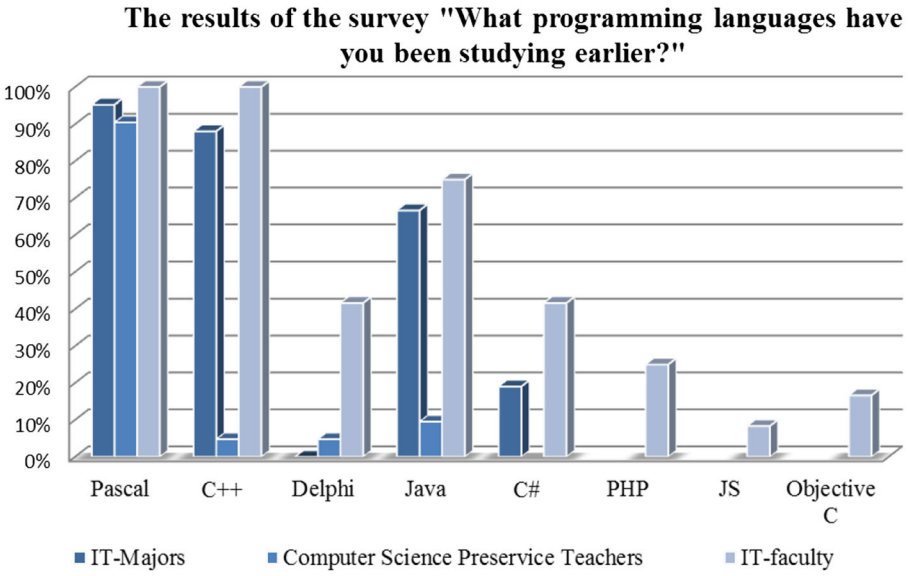


Fig. 3 The results of the survey «What programming languages have you been studying earlier?»

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 hands-on course «Basics of iOS apps development», the participants of the summer school learned how to develop mobile applications in the X-code environment in the Objective-C language. The following topics were covered in the course to lay the knowledge basis on mobile application development:

1. Basics of creating a Single View application.
2. The UI controllers.
3. The using the multimedia.
4. The using sensors in the application.
5. The programming complex View.
6. SQLite database.
7. Final project.

During in-class students learned how to develop mathematical models and algorithms for mobile applications using demo examples of mobile applications. During hands-on session the students created applications according to the instructions, tested and debugged the applications. To consolidate the skills under the individual work, students developed their own applications, working in small team. Such course organization allowed implementing a spiral model, which proposes scaffolding through enriching and enhancing developed mobile applications. The in-class structure is not a linear, it has a spiral structure. It means that the challenge set at the beginning of the in-class requires to be returned to it repeatedly during the in-class. During the course, students

developed projects from simple «Hi, Kazakhstan! « to a complete software with functions of search and update from the Internet.

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.

As can be seen from the table, computer science preservice 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. The reason for this is that in high school the programming is learned only in context of Pascal programming language and within the short course duration. In teaching such students, the lecturer needed more time for detailed explanation of the theoretical material and performing hands-on projects. 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 preservice 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.

IT-Majors and IT-faculty are more self-sufficient in performing hands-on projects and have proficient CT skills, which is consist in the ability to analyze and solve a problem, perform debugging, gain experience through analyzing demo examples, creatively perform individual tasks. A sufficient level of preliminary knowledge of programming languages among IT-majors and IT-faculty (76% and 97%) provided a good foundation for acquiring and adapting to new learning content of mobile development. Thus, during the experiment, we came to the conclusion that the level of basic programming knowledge influences the aquisition of a more complex content in the design of mobile applications. Moreover, the results of experimental training obtained during survey, testing and pedagogical observation confirm the availability several types of the knowledge gap: 1) the gap between the actual basic knowledge of high school students and the knowledge necessary for the successful mastery of university learning content corresponding to state-of-the-art computing; 2) the gap in the curriculum of first-year IT-Majors and computer science preservice teacher.

Table 1 The results of the achievement testing and evaluation application of course participants (average value)

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

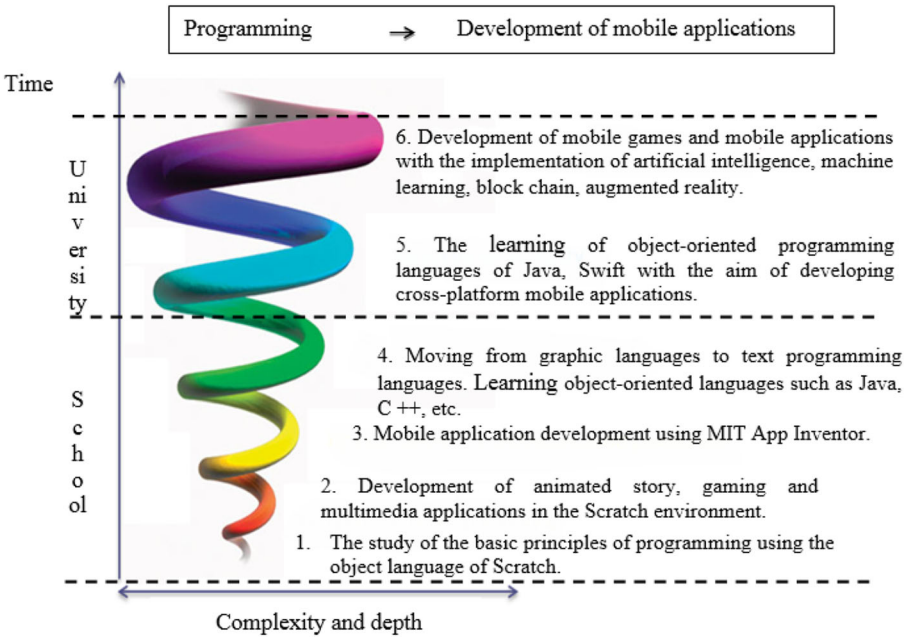


Fig. 4 Spiral model teaching mobile application development

In this regard, the computer science high school curriculum should provide better achievement of the basic programming principles, the basic concepts of object-oriented programming and the basic principles of mobile application development.

5 Conclusion

Having studied international experience on bridging the gap in students' knowledge at high school and university level, we can conclude that one of the possible approaches to solve the research question is to overhaul school and university curricula with a progressive enhancement and develop knowledge from basic programming to mobile application development. Due to this approach, the learning units of knowledge, established at the elementary level, are progressively enhanced from one educational stage to another and developing appropriate skills. In this regard, it is advisable to implement a spiral model teaching mobile application development, according to which learning content in high school and university should be planned in terms of the spiral principle (Bruner 1960). This model should be applied to all IT-Majors, including Computer science preservice teachers, who must master mobile technologies at a professional level in order to train and apply them in the educational process.

Considering opinions of researchers on necessity to bridging the knowledge gap through continuity principle in learning content (Stone 2019; Buitrago Flórez et al. 2017; Dodero et al. 2017; Zorana 2003; Siadaty and Taghiyareh 2007; Offir et al. 2003), motivation for learning programming in primary and secondary schools (Dillashaw and Bell 1985; Scherer et al. 2018; Kafai and Burke 2013; Inhelder and Piaget 1958; Yardi and Bruckman 2007), the efficiency of learning programming in

Scratch and further move to the mobile application development in App Inventor (Saltan 2016; Cheung et al. 2009; Maloney et al. 2010; Funke et al. 2017; Resnick et al. 2009; Morelli et al. 2011; Wagner et al. 2013; Karakozov and Manyakhina 2016), the moving from graphic to text programming languages (Cheung et al. 2009), the article proposes a spiral model, teaching the programming and mobile application development in high school and university in the context of the education system of Kazakhstan, which can be used in countries with similar education system (see Fig. 4).

The spiral model covers all levels of teaching programming from high school to higher education and suggests developing knowledge from introductory programming to mobile application development with the following educational path:

1. The teaching of the basic principles of programming using the object language of Scratch.
2. Development of animated story, game and multimedia applications in the Scratch environment.
3. Mobile application development using MIT App Inventor.
4. Moving from graphic languages to text programming languages. Learning the basis of object-oriented languages such as Java, C ++, etc.
5. The deeper learning of object-oriented programming languages such Java, Swift for the development of cross-platform mobile applications.
6. Development of mobile games and mobile applications with the implementation of artificial intelligence, machine learning, block chain, augmented reality.

The first four levels must be implemented in high school. Moreover, the emphasis is laid on mastering the basic principles of programming and developing mobile applications through graphic or visual programming environments such as Scratch, MIT App Inventor. The model also provides a level of moving from visual or graphical programming environments to higher-level text-based programming languages. The fifth and sixth levels of the model imply implementation at the university level and are focused on learning how to develop mobile games and mobile applications due to global trends of mobile computing.

The proposed spiral model teaching mobile application development is based on the study of international experience in the field of teaching programming and mobile application development in the high school and university and on the current state of the national system of teaching programming.

This model can be used in the development of curricula in computer science at high school and educational programs at university, as well as to build a methodological system for teaching programming and mobile application development at different levels from high school to university in order to train highly qualified in-demand mobile developers.

References

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. https://doi.org/10.1145/2380552.2380620

Araujo, L. G. J., Bittencourt, R. A., & Santos, D. (2018). An Analysis of a Media-Based Approach to Teach Programming to Middle School Students. *In Proceedings of The 49th ACM Technical Symposium on Computer Science Education*, Baltimore, MD, USA, Feb. 21–24, 2018 (SIGCSE '18), https://doi.org/10.1145/3159450.3159526

Bers, M. U. (2017). *Coding as a playground: Programming and CT in the early childhood classroom*. Routledge.

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

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching CT through programming. *Review of Educational Research, 87*(4), 834–860.

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.

Coelho, C. S., & Moles, D. R. (2016). Student perceptions of a spiral curriculum. *European Journal of Dental Education, 20*(3), 161–166.

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.* IN: Indiana Retrieved from https://files.eric.ed.gov/fulltext/ED255360.pdf.

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.

Fronza I, Corral L, Pahl C (2019). Combining Block-Based Programming and hardware prototyping to Foster CT. In *Proceedings of the 20th Annual SIG Conference on Information Technology Education* (pp. 55-60). ACM.

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.

Good, J., Yadav, A., & Mishra, P. (2017). CT in computer science classrooms: Viewpoints from CS educators. In *Society for Information Technology & Teacher Education International Conference* (pp. 51-59). Association for the Advancement of computing in education (AACE).

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. 10. 1037/10034-000.

Joshi, G., & Desai, P. (2016). Building software testing skills in undergraduate students using spiral model approach. *In 2016 IEEE eighth international conference on technology for education* (t4e) (pp. 244-245). IEEE.

Kafai, Y., & Burke, Q. (2013). Computer programming goes back to school. Phi Delta Kappan, 95(1), 61–65. https://doi.org/10.1177/003172171309500111. https://www. researchgate. net/publication/256005803. Accessed: 25 May 2019.

Karakozov S.D., Manyakhina V.G. (2016). Teaching informatics in South Korea: The analysis of textbooks for primary and secondary schools. *Informatics and Education* (1):11–16. (In Russ.)

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.

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).

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.

Offir, B., Barth, I., Lev, Y. & Shteinbok, A. (2003). Teacher–student interactions and learning outcomes in a distance learning environment. *Internet and Higher Education, 6*(1), 65-75. Elsevier Ltd. https://www.learntechlib. org/p/96514/. Accessed: 17 May 2019.

Papert, S. (1980). Mindstorms. In *Children, computers and powerful ideas*. New York: Basic books.

Papert, S. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.), *Constructionism*. Cambridge: MIT Press.

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. https://doi.org/10.1109/SECM.2017.12

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.

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

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.* Advance online publication. https://doi.org/10.1037/edu0000314.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying CT. *Educational Research Review, 22*, 142–158.

Siadaty, M., Taghiyareh, F. (2007) PALS2: Pedagogically adaptive learning system based on learning styles, icalt, (pp. 616-618), *7th IEEE International Conference on Advanced Learning Technologies (ICALT)*. https://doi.org/10.1109/ICALT.2007. 198. https://www. researchgate. net/publication/221424232. Accessed: 09 May 2019.

Standard curriculum on the subject (n.d.-a). «Computer science» for grades 5–9 of basic secondary education on the updated content. Developed in accordance with the State compulsory standard of secondary education (primary, basic secondary, General secondary education), approved by the Government of the Republic of Kazakhstan dated August 23, 2012 № 1080.

Standard curriculum on the subject (n.d.-b). «Computer science» for grades 10–11 of basic secondary education on the updated content. Developed in accordance with the State compulsory standard of secondary education (primary, basic secondary, General secondary education), approved by the Government of the Republic of Kazakhstan dated August 23, 2012 № 1080.

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

Taft, D. K. (2007). *Programming grads meet a skills gap in the real world*. Retrieved September.

Tan, P.H., Ting, C. Y., & Ling, S. W. (2009). Learning difficulties in programming courses: undergraduates' perspective and perception, *International Conference on Computer Technology and Development* 2009 (ICCTD'09), 42–46, 2009.

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

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Yardi, S. & Bruckman, A. (2007) What is computing?: Bridging the gap between teenagers' perceptions andgraduate students' experiences. In Anderson, R., Fincher, S., & Guzdial, M. (Eds.), *Proceedings of the Third international Workshop on Computing Education.* https://www. cc. gatech. edu/conferences/icer2007/slides/yardi-talk. pdf. Accessed: 11 May 2019.

Zorana, E. (2003). Bridging the knowledge gap between secondary and higher education. *College and Research Libraries, 64*, 75–85. https://doi.org/10.5860/crl.64.1.75 https://www. researchgate. net/publication/279437426.

Affiliations

G. Aimicheva[1] · Zh. Kopeyev[1] · Zh. Ordabayeva[2] · N. Tokzhigitova[3] · S. Akimova[4]

Zh. Kopeyev
zhanat_kb@mail.ru

Zh. Ordabayeva
zhamalika@mail.ru

N. Tokzhigitova
nurgul287@mail.ru

S. Akimova
saule_akim@mail.ru

[1] Department of Computer science, L.N. Gumilyov Eurasian National University, Nur-Sultan, Kazakhstan

[2] Senior lecturer at Pavlodar State University named after S. Toraigyrov, Pavlodar, Kazakhstan

[3] Pavlodar State University named after S. Toraigyrov, Pavlodar, Kazakhstan

[4] Senior lecturer at M.Utemisov West-Kazakhstan State University, Uralsk, Kazakhstan