ACTIVE LEARNING METHODS IN PROGRAMMING FOR NON-IT STUDENTS

Olga Mironova, Irina Amitan, Jüri Vilipõld and Merike Saar
Faculty of Information Technology, Department of Informatics,
Chair of Software Engineering, Tallinn University of Technology
Akadeemia tee St. 15A, Tallinn 12618, Estonia

ABSTRACT

The purpose of this study is to demonstrate a teaching approach and some teaching strategies in an Informatics course for the first-year non-IT students at the Department of Informatics of Tallinn University of Technology, Estonia. The authors suggest some solutions for making the course, which is usually complicated, more dynamic and attractive, thereby raising students' interest and motivation with the aim of achieving the set learning outcomes.

KEYWORDS

Programming basics, non-IT.

1. INTRODUCTION

In recent years information technology is rapidly developing and playing a great role in contemporary life. This role is equally important at work or study and at home. Consequently, students have to grasp different computing skills to feel confident and be ready for the future work and able to participate effectively in the IT-world. Therefore, the main objective of modern computer education for the non-IT students today is to teach them to be always well-informed in technical fundamentals and be able to find a common ground with IT-specialists at their future workplace.

In the current work the authors try to find answers to questions about how to achieve better results in the teaching of programming basics for the non-IT students who are beginners in this field.

2. PROGRAMMING BASICS

2.1 The Informatics Course

Computer education basics have been included into all the curricula at Tallinn University of Technology (TTÜ, 2016) and have been consolidated for all non-IT specialities into a course named "Informatics". The named course lasts for two semesters and students have two academic hours weekly at computer classes. Usually the students' group size is approximately 25 students; this number annually depends on the total number of the matriculated students.

During the course lecturers apply modern and classic face-to-face classroom methods, pair or group work and provide students with independent learning in the Moodle e-environment (Moodle, 2016).

The first Informatics module is informational work such as text processing, presentations and spreadsheets handling. The work tools here are MS Word, MS PowerPoint and MS Excel. In addition, we use Google tools as an alternative to the Microsoft products.

The programming basics part of the Informatics course lasts for one semester and it starts with Scratch (Scratch, 2016). Scratch is a graphical programming environment, which is very intuitive and greatly helps learners to take on board the main concepts and terms of modelling and programming such as the data,

process, branching or iteration. Such visual programming lasts for 5 or 6 weeks, depending on the students' level in each group.

The next course module is dedicated to Visual Basic for Applications (VBA) (Sissejuhatus VBAsse, 2016) or Python (Python Software Foundation, 2016), (Tutvumine Pythoniga, 2016). VBA choice is reinforced by the already familiar environment MS Excel – it is one of the Informatics first module topics in the Informatics course. In addition, it is comfortable for beginners to keep their data in the same fail with the programme. Python does not require any declaration of simple variables, which makes working with it easier for the beginners.

The main learning outcomes of the Informatics course are introduced below (Õppeinfosüsteem, 2016). A student who completes the course:

- acquires the foundations of problem analysis and system modelling.
- analyses relations between objects and provides rationale for the algorithms and methods applied.
- is familiar with the nature of data and objects and can specify them and use them in programs.
- is familiar with and describes, using VBA/Python and UML activity diagrams, the main activities occurring in programs and algorithms.
- is familiar with the nature and main concepts of object-oriented programming.
- composes programs consisting of multiple procedures and organizes the data flow between them.

The course aims at reaching the results in two different but tightly linked ways: learning to understand the object-oriented approach and getting the necessary skills in building algorithms. Both skills have to be implemented in simple applications.

It should be mentioned that the programming part of the Informatics course seems to be rather complicated for most of the non-IT students. The course authors and teachers face some problems. The biggest of them is lack of preparation and lack of prior knowledge in programming among the first-year non-IT students. As a result, learners immediately lose their motivation at the beginning of the programming part of the course. This deplorable fact, in its turn, leads to poor knowledge and poor academic results. Consequently, the authors of the course try to improve the program and content from year to year with the aim of finding the best solutions to achieve the goals and fully get the outcomes (Robins et al, 2003), (Kak, 2014).

2.2 Modelling and Algorithmization

In the programming module of the Informatics course we try to focus mostly on the models and algorithms. Our aim here is to teach students how to build a model and an algorithm of the problem and describe it using both familiar and new tools.

For these programming tasks, teachers and students together try to build UML (Unified Modelling Language) activity diagrams to describe the algorithms. It is the first step in any task solution. In addition, a verbal description and a pseudo code are typically used.

As was mentioned above, we start the programming practice with Scratch. At this stage students have to understand the problem and be able to describe it. Syntax errors are impossible in Scratch and this fact gives students an opportunity not to think about mistakes but concentrate only on the model and the algorithm. Moreover, Scratch graphical blocks give a holistic and lively picture of the created model.

After creating the Scratch projects as an introduction to programming, the course instructors can also use its scripts to visualize, formulate and describe the problem. Thus, we get one more opportunity to introduce to our students a problem to be solved – providing them with a Scratch script.

It should be mentioned that the course teachers do not give students volume tasks for solving. Our main idea is to explain through small tasks why and how it works.

2.3 Active Teaching and Learning

To raise and keep students' motivation in the learning process it is necessary to make the routine learning process more attractive and dynamic for them. In the current section of the paper the authors suggest some ways for making programming more engaging for non-IT learners with the aim of raising their interest.

2.3.1 E-learning

As was mentioned above, during the semester all students work in the Moodle e-environment. This independent work mostly consists of grasping the new material, taking self-tests and other learning tasks.

For better understanding, the theoretical material should be presented to non-IT students in simple and clear forms. A multitude of books and any other materials provided for them does not mean that they are useful for the students; moreover, this can be intimidating for the beginners.

During the Informatics course development and evolution its authors have been trying to adapt theoretical teaching materials for the e-environment and deliver it to students in most suitable forms. Our conclusions here are to provide students with small portions of the learning material and maximally visualize them. In this context, a small portion does not mean that students will lack some knowledge – it means that they are provided with well-filtered materials, which are adapted to non-IT beginners. A great role here is played by short teaching videos that explain the main programming concepts and usage cases. The same principles have been used in the Khan Academy (Khan Academy, 2016).

After each new topic students have to fulfil corresponding tests. Modern testing systems provide us with a variety of test types and, using them the course authors have worked out a test system which uses tasks similar to the pre-prepared program tasks. As experience shows, the most effective type of the test, which greatly helps non-IT students, is "fill in the gaps" type. Students get the problem description and the corresponding program text with some gaps and their task is to fill in gaps and check their results afterwards. Doing this, students learn the syntax and learn to understand the algorithm. In addition, tests where students have to make the program text out of sentences and arrange them in the correct order, teach students to see and understand a model of the proposed tasks.

It is very useful for the students to check and correct their classmates' programmes. This activity develops such an important skill as the ability to read and understand a code written by another person. The Moodle environment provides us with this opportunity and we periodically use it in the course.

It should be mentioned that students' independent work in Moodle is divided into stages, which are ordered and a transition to the next stage is possible only after finishing the previous one. Such course construction helps the course teachers to monitor the current situation in students' knowledge. Moreover, such a system brings a competitive note into the learning process and makes it more attractive.

2.3.2 Face-to-face lessons

It should not be expected from non-IT students and especially from the beginners that they immediately start to write a program code after the first explanations. The best option here is a simple copying task from the teacher's screen projected on the board. During this copying students do not think about why it is so, their interests are limited to the fact that they need to copy some text on time and afterwards check whether the program works. It is great if they are able to do it on their own, however, usually students are not able to check and correct it due to incomprehension of the solution. To make the situation more positive, our group of teachers have worked out some strategies that can help to involve students in the coding process during face-to-face lessons.

Firstly, it should be mentioned that during the programming module of the course, the majority of the created applications are small games that students can play. Using Scratch, students make games and within the process, learn to understand their algorithms (Rakenduste loomine Scratchiga, 2016). Scratch gives an opportunity to test and, if necessary, correct the algorithm immediately without thinking about the syntax. Afterwards, when an algorithm is already clear, it is simple to translate it to VBA or Python, concurrently learning the syntax. Thus, a teaching tool like Scratch already adds an element of attractiveness to the course.

Secondly, as a group-work assignment during a face-to-face lesson it is possible to offer students a possibility to play a game: they have the algorithm of a program and each student in the class should write one line in the code. The named method is quite controversial, but it is useful at the beginning of coding, when students just learn the basics. Afterwards, when each student has his/her own programming style, it is not so useful. However, it teaches to understand others' manners and proves and demonstrates why one solution can be better and more logical than another. It should be noted that this is important knowledge in any subject, not only in programming. In addition, this game greatly helps teachers to maintain a high level of students' attention during the lesson.

The next assignment for students, which we successfully apply, especially before practical tests, is correction of mistakes in a programme text. As usual, the students have their task descriptions and a ready program, which does not work at all or does not work properly. The number of mistakes is also known. The mistakes are different: from simple misprints to syntax or logical errors. As our experience shows, such assignments are useful if offered as pair work. Here we also develop group work skills among our students and teach them to understand the programme code. Besides, in such a way students learn the syntax by discussing it.

In addition, compared to the previous learning task, the new system works effectively. This is a bonus. During the semester students can collect the bonus points and, if necessary, use them at the final exam. Students can get these points, for example, for solving some additional tasks in the lesson, at home or in the Moodle environment. Usually the bonus system is determined at the beginning of the semester and sometimes it is an additional reason for students to attend the lesson.

When new theoretical material is explained, the course teachers often use ready-made programmes to introduce some topics to learners. In such a case, it is advisable to prepare, in advance, some mistakes in the code and within the discussion, correct them together with the students. This way, new concepts are assimilated much better and students learn to respond to different types of errors and afterwards are not afraid of them.

The teaching strategies mentioned above are the main types that we apply in the Informatics course and they are aimed at raising students' interest in the programming subject by better engaging them into the learning process. As students' feedback shows, these methods work and bring positive results. It is necessary to apply different strategies in teaching, not only in programming, with the aim of varying students' learning and educators teaching experience and style.

3. CONCLUSION

Based on the above said, it should be concluded that the authors of the Informatics course for the first-year non-IT students focus mostly on the model, algorithm and also their visualization, rather than teaching syntax and coding techniques (Vilipõld et al, 2013). The course authors consider that active learning greatly helps not only students to grasp new knowledge and achieve better results but it helps teachers to develop, improve and raise their pedagogical skills to a new level.

In the future development of the Informatics course, the authors try to keep up with times and main trends in pedagogy and computer education (George Lucas Educational Foundation, 2014), (George Lucas Educational Foundation, 2014).

REFERENCES

George Lucas Educational Foundation, 2014. Coding in the Classroom. http://www.edutopia.org/topic/coding-classroom

George Lucas Educational Foundation, 2014. Project-Based Learning. http://www.edutopia.org/project-based-learning

Kak, A., 2014. Teaching Programming. https://engineering.purdue.edu/kak/TeachingProgramming.pdf

Khan Academy, 2016. http://www.khanacademy.org

MIT Media Lab, 2016. Scratch. http://scratch.mit.edu/

Moodle – Open-source learning platform, 2016. https://moodle.org/

Õppeinfosüsteem, 2016. http://ois.ttu.ee

Python Software Foundation, 2016. https://www.python.org/

Rakenduste loomine Scratchiga, 2016. http://rlpa.ttu.ee/scratch/Rakenduste_loomine_Scratchiga.pdf

Robins, A., Rountree, J. & Rountree, N., 2003. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13(2), pp. 137-172.

Sissejuhatus VBAsse, 2016. http://rlpa.ttu.ee/vba/VBA.pdf

Tallinn University of Technology, 2016. http://www.ttu.ee/

Tutvumine Pythoniga, 2016. http://rlpa.ttu.ee/python/Python.pdf

Vilipõld, J., Antoi, K., Amitan, I., 2013. Rakenduste loomine ja programmeerimise alused. Valikkursus gümnaasiumitele. http://rlpa.ttu.ee/RLPA_opik.pdf