Teaching Computing for non-IT students

Experience of Tallinn University of Technology

Olga Mironova, Jelena Vendelin, Irina Amitan,
Jüri Vilipõld, Merike Saar
Department of Informatics
Tallinn University of Technology
Tallinn, Estonia
{olga.mironova, jelena.vendelin, irina.amitan,
juri.vilipold, merike.saar}@ttu.ee

Tiia Rüütmann
Department of Industrial Psychology
Tallinn University of Technology
Tallinn, Estonia
tiia.ruutmann@ttu.ee

***Abstract* – the present paper demonstrates the teaching methods and principles in informatics course for the first year non-IT students at Tallinn University of Technology, Estonia. The authors suggest some ways for active learning and achieving better results in the issues that are usually complicated for the beginners.**

Keywords—computer science; informatics; teaching methods; Scratch; Pyhton; VBA; algorithm; program; non-IT

I. Introduction

Every area of life today is powerfully affected by the explosion of new information technologies, robotics, biotechnology and the increased blending of invention with scientific discovery. What professionals know and can do are critical resources for the society today.

The purpose of present-day engineering education is to provide learning, which is required by students to become successful specialists or educated workers with technical skills, social awareness and knowledge of innovation. A combined set of knowledge and attitudes, based on technologically complex and sustainable products, processes and systems, is essential to strengthening productivity, entrepreneurship and excellence in the society. Accordingly, we should improve the quality and nature of education. Thus, the objective of engineering education today is to educate students to be knowledgeable in technical fundamentals.

II. The Informatics Course Description

The Informatics course, designed for the first year non-IT students is a responsibility of the Institute of Informatics at Tallinn University of Technology.

The aim of the course is to develop logical, analytical and algorithmic reasoning skills as well as the ability of investigating problems and tasks in a systematic way. This is what is tried to achieve through two main parts of the course.

The duration of the course is one or two semesters, depending on the speciality of the students. The course is taught in three languages: Estonian, English and Russian.

As the first stage, the students learn to create applications with standard general-purpose office software; the second part of the course is devoted to the basics of programming. During the whole course the instructor is guiding the learner to solve problems using the object-oriented approach.

A. The First Part. Applications Using General-Purpose Software

The learning process starts with creating documents and presentations. It is followed by spreadsheet processing, where students deal with formulas, diagrams, built-in functions and facilities. The aim of this part is to get familiar with the large amount of software products and find out the best tools.

This comparatively simple part provides students with deeper skills than needed for an average white-collar work. Students create applications which have to be flexible and reusable. Creating styles, master documents and templates enables to demonstrate and practise document and presentation design. In spreadsheet applications, the principles of data processing are introduced, using data tables as an implementation of object classes. All the topics are shown using MS Office programmes, but the students are encouraged to use the same principles to solve tasks using any other software. The duration and contents of different modules depend on the field of study of each students and is coordinated with their curriculum.

B. The Second Part. Programming

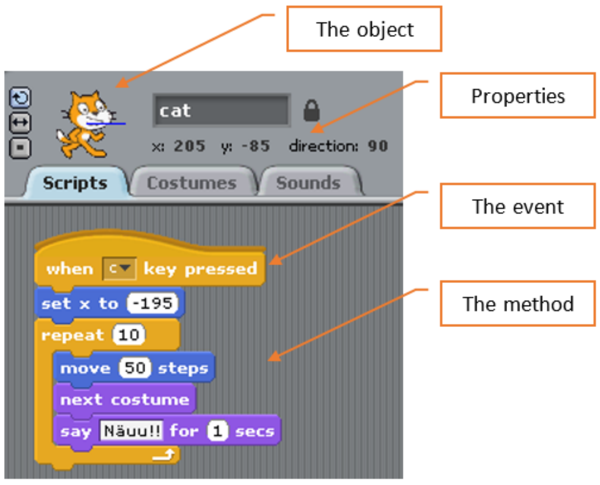
In the second part of the course the basics of programming in practice and the main principles of algorithmization are introduced. This part combines the object-oriented approach of solving problems with algorithmization and functional programming. The main goal of this part is to teach logical and algorithmic thinking, as well as systematic investigation.

It should be noted that the programming part of the Informatics course is usually more complicated for most of the students. The difficulties in teaching programming have been described and systematized [1], [2]. However the problem remains. Therefore, the contents of this part varies from year to year and from speciality to speciality, in the attempt to find the best environment to fulfil the goals.

After a few years of practice we came to the conclusion that a graphical environment such as Scratch [3] is an effective introductory tool for understanding both the object-oriented approach and functionality of a programme.

Scratch is not designed to solve complicated tasks, but it is simple, very expressive and makes understanding the objects a game.

Creation of objects is solved by importing or drawing graphics; combining the blocks for each object creates methods for these objects. Some of the blocks are used to show the reaction of the object to some events. We see here the main aspects of object-oriented programming resulting in an attractive animation (Fig. 1.).



```
when c key pressed
set x to -195
repeat 10
    move 50 steps
    next costume
    say Näuu!! for 1 secs
```

Fig. 1. A fragment from the application in Scratch

The next step is to proceed with more serious tasks in a way which could be useful for the students in the future.

The solved problem has to be analysed and described in a model. Building the models for applications is supported by diagrams in UML (Unified Modelling Language). It helps the learners to reach the solution without reference to any of the programming languages. At the beginning, the teacher provides the students with a prepared model, which is analysed in a group. The analysis is followed by writing the programme code according to the diagrams. Later on, the students have to create the models themselves.

Different environments and programming languages have been used during the years. At the moment, two main streams are used: Python and Visual Basic for Applications (VBA).

There are several arguments to prefer one over the other.

Python supports structured programming and procedural styles. In addition, Python does not require the declaration of the simple variables, which makes work with it easier for the beginners. It has a large and comprehensive standard library. Python interpreters are available for installation on many operating systems, allowing Python code execution on a majority of systems. It is an open source and is available to all students. The language is a high-level language and its syntax allows programmers to express concepts in fewer lines of code than would be possible in some other languages. There are a lot of tutorials and visualized debugging tools available. It makes it possible to provide learning support in different ways and everyone can find the most suitable one for himself. On the other hand, the huge amount of information is often confusing and students need detailed guidelines from the instructor.

Python supports the object-oriented approach, but a lot of work could be done without it. It seems to be rather complicated for the beginner to orient in the documentation of the built-in classes and classes from different libraries. In our introductory course we usually do not include creation of classes and use only some of the existing objects. Therefore, the object-oriented approach is not taught and is limited to possibilities provided by Scratch.

Visual Basic for Applications as well as Python support structural programming and provides the programmer with many built-in facilities. The text editing and debugging tools for the programme code are visual and well observable.

However, using VBA cannot go far without introducing objects. There are many classes in the standard environment that can be used with their properties and methods.

VBA procedures are attached to the applications, so there are no interface problems (as the interface is included in the application itself). On the other hand, one has to use the object classes of the application for any input or output. However, some of the objects have a complicated structure and relationships with other classes, which is confusing for the majority of the beginners.

It was found that using MS Excel for application creation is the easiest and most understandable way to practice using classes. This is one of the reasons why MS Excel has been chosen as an environment for creating applications with VBA. The classes of worksheets and cells are comparatively easily used. The expressions and many of the built-in functions in VBA have similar names and arguments with those in Excel. These topics are covered in the textbook [4].

C. *Teaching School Teachers and Pupils*

It was found that secondary school students are very differently prepared for the university Informatics course. There is no common programme or any teaching system for computing subjects in Estonian schools; some of the schools do not teach them at all.

Several classes covering some topics of our Informatics course, mostly including programming, are organized both to teachers and pupils of secondary and elementary schools. Some of the materials were prepared specially for these classes, such as [5], [6] and [7].

The aim is to introduce the idea of the university course to teachers and to motivate them to pass the knowledge on to their pupils. The classes for pupils try to introduce programming in the form of a game and prove that it is not that complicated as most of them believe it to be.

With these courses for school pupils and teachers we provide better preparation for university.

III. Teaching Strategies

A. The Principles

One of the main problems concerning the first year students is that they are not enough prepared to study on their own as expected at university.

Another one is lack of motivation, because the first year students have usually little idea of what knowledge and skills they will need in future.

Trying to reduce the difficulties in the learning process and improve motivation, we combine different methods of active learning and teaching throughout the whole course [8], [9], [10]. Classic face-to-face classroom methods, group work and learning in the Moodle e-environment are used [11].

We find it very attractive to use the project-based method of learning. In fact, it has better results with students who have previous working experience, and is more difficult to implement with students who have just recently finished a secondary school (as they do not see any links with real life). However, we try to find motivating tasks as well as link our activities to other courses, such as Chemistry, Mathematics, Physics or Business.

B. The Organization of Learning Process

The contacts with the instructor take place once a week and last up to two or three academic hours. During this comparatively long time students are kept active and motivated. To achieve it, different styles of active teaching and learning are applied.

The lesson usually starts with a short test on previous topics. In the middle part the students start solving a new or continue solving a problem started last time. At the end, students are asked to provide feedback with conclusions on how easy or difficult the lesson was, and whether they understood the purpose of the things taught.

C. The Testing System

To understand which level of difficulty we should apply in our course, we have to know the students' level of preparation for the subject. To get this information, we hold a test at the beginning of each semester. This test is based on alleged knowledge, which is necessary for studying Informatics. In accordance with their level, the students then get their learning materials and assignments.

The same test is held at the end of each semester. Last year's statistics give us an opportunity to evaluate the effectiveness of the applied the teaching methods [12].

At the beginning of each lesson the students take a short test to check their understanding of the material from previous lessons. This test does not take much time (approximately 5 minutes) but it gives students a stimulus to open the material for at least once before each lesson. As a result, students have continuous learning throughout the course.

At the end of each lesson we ask anonymous feedback about the new material and the lesson organization. This feedback gives us an overview of student satisfaction. Because we try to change our approach to teaching computer sciences, this kind of feedback is very useful for us. Due to its periodicity, we continuously monitor the learning process as well as students´ progress.

D. The Teaching Methods

We try to prepare students for effective use of the standard computer equipment for their speciality.

One of our problems related to teaching computer science is the place of our subject in the university curriculum. First year students do not have to solve real issues or tasks related to their speciality. Therefore, they do not need to use computers for learning and, as a result, they are not motivated to learn computer science. Our aim is to show how they can raise the effectiveness of their learning and work through the correct use of computers. In addition, it is very important to teach students how they can use the created algorithms to solve problems in their future work.

In teaching we use tasks where students have to make a project containing an analysis of initial data and data structure; the results and their structure as well as the methods that are necessary for the analysis. For example, students of the civil engineering faculty make projects like “Covering the floors” and “The calculation of the amount of the material for the detail”. For these projects students need different data from different resources: sizes, materials, prices, etc. Students of chemical faculty make projects related to laboratory work: calculations, reports, diagrams, etc. Besides the analysis and development of the application students have to present a formatted description of their project and a user manual.

Further in the course, the students develop and automatize these applications using the elements of programming.

In class, we often use pair work: one of the students begins to write the programme, the second finishes it. Usually we make those pairs at random. The aim of this division is to show students that a correctly written programme gives them an assurance in correct continuation, the best results and economy in time.

This method came from the situations when students claimed that their work is finished, the application is ready, it works and it does not matter how it was realized. It was difficult to explain that when the application works it does not mean that the code is correct. When we did the pilot tests of this type of pair work with our students they understood our approach.

In addition, we have to demonstrate to students that the testing of the application is a significant stage. During the testing they have to check different initial information to get the critical values. When these values are identified, the students define how the application should react to a situation. For example, students have to make sure that the created application does not only give an error message and close, but should direct the user to make the right changes.

E. Vizualizing of Programme Execution

To better understand and obtain knowledge in programming, it is very important to visualize the execution of the created program/application. It helps students to understand what is going on when the programme starts and works.

Visualization in Scratch helps students, who put the blocks together, to immediately see the results. They do not have to remember any syntax. They can concentrate only on the algorithms. It is very easy to show the parallelism and sequence of the processes using Scratch. In addition, the programme gives an opportunity to follow the relations between the objects using the mechanism of the messages.

Therefore, Scratch is an excellent way for learning algorithms and object-oriented programming.

For the visualization in VBA we use the debug mode during the programme execution. Students follow the programme step by step and fix their mistakes faster.

The Python's debugging system is more complicated, so for our students we usually use an internet application that was created for the visualization of the Python programmes [13]. Using this application students can see how their code is working and what its input and output is.

F. Not E-learning but E-support

We do not agree with the theory of "e-learning only" for our course. Face-to-face lessons give students an opportunity to communicate and get necessary skills for group work with colleagues. Sometimes we give students the opportunity to choose a team or form a pair, but sometimes the teacher makes the teams or the team is organized randomly. We try to prepare our students to work in situations where each member gets his/her own part in a project.

In addition, in the e-environment we make and use short videos to demonstrate algorithms and programme execution for all three environments: Scratch, VBA and Python. The same principles are used in Khan Academy [14].

We strive to raise the motivation, as fast as is possible. In addition, we try to create the connection between our Informatics course and other subjects.

IV. IT AND NON-IT

In their future work non-IT people have to communicate with IT-specialists. The joint project's success mainly depends on this communication. One of our aims is to teach non-IT students to talk with IT-department using the language which is clear and understandable for both sides.

As non-IT specialists are involved in many projects, we hereby list the basic phases of the application creation and define where non-IT specialists' participation is possible or/and necessary.

The first stage of application creation is the problem formulation. Non-IT specialists have to define the main functions, requirements and circumstances of the application to be created. Clear needs formulation by non-IT specialists helps to save time. They need to know how to explain their ideas or/and problems to their IT-colleagues. For this explanation non-IT people need to possess knowledge in computer terminology and use it correctly. For effective collaboration, subject specialists and IT-specialists should speak the same language. The first know what they need, the last are good at how to execute it. During their collaboration the circumstances and criteria are specified.

As mentioned above, non-IT students have learned and can use the UML schemes to illustrate their needs (in solving their issues) not only in words but also graphically.

The second step is the analysis. In this stage non-IT and IT-specialists work together. They have to define data, formulas, connections, methods, models, means of implementation, costs and time. If non-IT understand the basics of the project analysis from this point of view, constructive dialogue between both sides is possible. This dialogue is supported by the use of correct terms.

At the stage of application design, non-IT specialists help to create an interface and therefore they have to understand the data and programme structure, as well as algorithms.

During the design stage the application processes are identified. The process can be consecutive, parallel, repeating and brunching. For their realization, different algorithm elements are used. Scratch with its blocks suits well for learning the terms of cycle and brunching.

In the realization/implementation stage, the non-IT specialists help to debug the created application and check the design.

The next step is the launch of the application. Usually, the initial data input is the responsibility of non-IT specialists. When they clearly understand the main principles of application execution, they become more effective working with it.

Also, the application development is not possible without non-IT specialists. Thus, knowing the algorithms and terms proves helpful throughout their job.

V. SUMMARY

In their work process, non-IT specialists frequently use IT: they are involved in the formulation of the problem, drawing algorithms and design. Therefore, for the effectiveness of this collaboration mutual understanding has to take place between these two sides, IT and non-IT. Based on this and our experience in teaching computing, we have formulated the aims of the Informatics course.

They are:

- To teach students to organize and analyse information and then represent it through models.
- To teach to formulate the problem in a way that enables to use a PC to help solve it.
- To teach to apply algorithmic thinking in their task solutions and afterwards to automatize them.
- To teach generalization and transferring the problem solving process to other issues.

Trying to achieve our goals, we successfully apply a variety of teaching methods and techniques.

During contact lessons, students prefer group work, which gives them an opportunity to try the obtained knowledge in practice and get the support not only from the teacher. In addition, such kind of learning activities develop teamwork skills, especially among the first year students. This way, the role of the lecturer is a little different – we become advisors, motivators and supporters in the students' learning process.

E-learning in Moodle provides students with a large number of exercises, self-tests and a variety of different kinds of educational materials.

The practical tasks are adapted to students' specialization: economics, social, chemistry and civil engineering. We attempt to create more connections between our Informatics subject and the students' specialty. Feedback shows that students get an understanding of the Informatics subject and its applicability in their future study and work.

To sum up, it should be mentioned that problem solving methods that we use in teaching help students in their learning not only in computer subjects. When they are able to organize data logically and analyse the information afterwards, they can formulate any problem in any discipline better and more correctly. In its turn, a correct formulation of a task paves the way for its successful solution.

REFERENCES

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

[2] Kak, A., 2014. Teaching Programming. Available at: https://engineering.purdue.edu/kak/TeachingProgramming.pdf

[3] MIT Media Lab, 2013. Scratch. Available at: http://scratch.mit.edu/.

[4] Sissejuhatus VBAsse. Available at: http://rlpa.ttu.ee/vba/VBA.pdf

[5] Vilipõld, J., Antoi, K., Amitan, I. Rakenduste loomine ja programmeerimise alused. Valikkursus gümnaasiumitele. Available at: http://rlpa.ttu.ee/RLPA_opik.pdf

[6] Rakenduste loomine Scratchiga. Available at: http://rlpa.ttu.ee/scratch/Rakenduste_loomine_Scratchiga.pdf

[7] Tutvumine Pythoniga. Available at: http://rlpa.ttu.ee/python/Python.pdf

[8] Felder, R. M. & Silverman, L. K., 1988. Learning and Teaching Styles in Engineering Education. Engr. Education, 7(78), pp. 674-681.

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

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

[11] Moodle Trust, n.d. Moodle. Available at: https://moodle.org/.

[12] Mironova, O.; Amitan, I.; Vendelin, J.; Saar, M.; Rüütmann, T. (2014). Strategies for the Individualization of an Informatics Course . In: Annals of Computer Science and Information Systems: Federated Conference on Computer Science and Information Systems, September 7–10, 2014. Warsaw, Poland. IEEE, 2014, 835 - 840.

[13] Learn Programming by Visualizing Code Execution Available at: http://www.pythontutor.com

[14] Khan Academy. Available at: http://www.khanacademy.org