THESIS ON INFORMATICS AND SYSTEM ENGINEERING C130

Combination of Pedagogical Strategies and Teaching Techniques for Teaching Computer Science Basics to Novices

OLGA MIRONOVA

TTÜ PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

This dissertation was accepted for the defence of the degree Doctor of Philosophy in Computer Science on June 15, 2017.

Supervisors: Associate Professor Tiia Rüütmann
School of Engineering,
Department of Mechanical and Industrial Engineering
Tallinn University of Technology
Tallinn, Estonia

Associate Professor Enn Õunapuu
School of Information Technologies,
Department of Software Science
Tallinn University of Technology
Tallinn, Estonia

Opponents: Professor Dana Dobrovská
Department of Pedagogical and Psychological Studies
Czech Technical University in Prague

Associate Professor Pedro Isaias
Institute for Teaching and Learning Innovation
The University of Queensland, Australia

Defence of the thesis: October 18, 2017, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for any academic degree.

Olga Mironova

ISSN 1406-4731
ISBN 978-9949-83-155-5 (publication)
ISBN 978-9949-83-156-2 (PDF)

INFORMAATIKA JA SÜSTEEMITEHNIKA C130

Pedagoogika strateegiate ja õpetamise tehnikate kombinatsioon informaatika algkursuse õpetamisel algajatele

OLGA MIRONOVA

TTÜ
KIRJASTUS

Table of Contents

LIST OF PUBLICATIONS

All publications are reprinted in the appendixes of the thesis.

The work of this thesis is based on the following publications:

A Mironova, O.; Rüütmann, T.; Amitan, I.; Vilipõld, J.; Saar, M. (2013). Computer Science E-Courses for Students with Different Learning Styles. In: Annals of Computer Science and Information Systems, 1: Federated Conference on Computer Science and Information Systems, September 8–11, 2013. Kraków, Poland. IEEE, 735−738.

B Mironova, O.; Amitan, I.; Vendelin, J.; Saar, M.; Rüütmann, T. (2014). Strategies for the Individualization of an Informatics Course. Annals of Computer Science and Information Systems, 2: Federated Conference on Computer Science and Information Systems, September 7–10, 2014. Warsaw, Poland. IEEE, 835−840.

C Mironova, O.; Amitan, I.; Vendelin, J.; Vilipõld, J.; Saar, M. (2016). Maximizing and personalizing e-learning support for students with different backgrounds and preferences. Interactive Technology and Smart Education, 13 (1), 19−35, 10.1108/ITSE-09-2015-0025.

D Mironova, O.; Amitan, I.; Vendelin, J.; Vilipõld, J.; Saar, M. (2015). Object-Oriented Programming for non-IT Students: Starting from Scratch. International Journal of Engineering Pedagogy, 5 (4), 22−28, 10.3991/ijep.v5i4.4734.

E Mironova, O.; Amitan, I.; Vilipõld, J.; Saar, M. (2016). Active learning methods in programming for non-IT students. Proceedings of International Conference on e-Learning 2016: 10th International Conference on e-Learning; Funchal, Madeira, Portugal; 1 – 3 July 2016. IADIS Press, 239−242.

AUTHOR'S CONTRIBUTION TO THE PUBLICATIONS

In articles mentioned below the main author is the author of present thesis.

A The author's contribution was the idea and the implementation of the model of the Computer Science Basics course.

B The author's contribution was the development of the course model and teaching approach. In addition, the author was responsible for data collection and analysis, computations and writing the publication.

C The author's contribution was the development of chosen teaching approach and data and analysis.

D The author's contribution was the development of the idea of the teaching approach to object-oriented programming for non-IT students.

E The author's contribution was the revision and systematization of learning techniques, which are used in programming module of the Computer Science Basics course.

INTRODUCTION

E-learning and blended learning [1] are rapidly developing world-wide systems. Currently, it is not possible to imagine any educational process without e-components or a holistic e-learning system. The main aim of such systems is to provide quality knowledge in a convenient form for its consumer – a learner.

At the same time, abundance of educational information and learning materials does not guarantee perfect knowledge. Educational materials should be carefully structured to cater for learners' needs and preferences with the aim to provide them with quality knowledge and guarantee the learning success.

All present-day knowledge in education is changing so fast that educators cannot predict what the 21th century students will need to know tomorrow. Instead, teachers should be helping to develop learning skills and strategies so that learners will be able to learn whatever they need to.

A combined set of knowledge, skills and attitudes is essential to strengthen productivity, entrepreneurship and excellence in an environment, which is based on technologically complex and sustainable products, processes and systems. Similarly, educators could improve the quality and nature of education.

Thereby, the objective of education today is to teach students more effectively using fundamental knowledge and modern techniques.

In present study the author aims to demonstrate that maximizing and personalizing learning support, which is composed of successful teaching strategies and techniques, greatly helps students with different backgrounds and preferences to reach the high level of knowledge.

Motivation for the Study and Research Object

Nowadays Computer Science is an imprescriptible part of any curriculum, and its content develops and changes fast. The general aim of this science is to develop logical, analytical and computational thinking by using computer on the highest level in any field of its applying.

One of the central places in modern computer education takes the concept *Computational thinking* [2, 3], which embraces:

- the ability to think algorithmically;
- the ability to think in terms of decomposition;
- the ability to think in generalisations, identifying and making use of patterns;
- the ability to think in abstractions, choosing good representations;
- the ability to think in terms of evaluation [4].

Computational thinking skills enable pupils to access parts of the Computing subject content. Importantly, they relate to thinking skills and problem solving across the whole curriculum and though life in general. Computational thinking is a series of thinking activities of human being who apply the methods and theories of Computer Science to simplify, insert, transform and simulate problems [5]. Computational thinking includes skills such as conceptualizing at multiple levels of abstraction, defining and clarifying a problem by breaking it down into relational components, and testing and retesting plausible solutions [6].

Based on contemporary literature, it can be argued that Computer Science teaching and learning has been considered to be difficult [7]. There are a number of reasons for this. Firstly, Computer Science subjects involve complex conceptual understandings, the acquisition of basic knowledge, highly technical terms and problem-solving skills. Secondly, students need to have proficiency in technical and practical skills with a range of hardware and software. Finally, Computer Science courses must include strong collaborative learning opportunities between students, their peers and teachers to develop problem-solving skills and apply complex theory to practical applications [8].

In addition, some challenges are related to teachers' own difficulties or the fact that the students have difficulty in understanding the material and in problem solving [9].

According to the author's opinion, basic challenges, which learners and teachers can face, are clearly classified and introduced in [10].

It should be certainly noted here that named sources also suggest ways to solve named issues. All these techniques successfully work and bring positive results.

Having reviewing the literature and on the basis of colleagues' and author's own experience in Computer Science Basics teaching the author should mark that new approach to teaching computing is needed. The novelty of the present thesis is the combination of two facts in teaching: *"what learners know before me"* and *"how they learn with me"*. The second concept in this approach is programming basics teaching technique. This technique is based on visualization and building algorithms rather than syntax and uses the first programming language as a support in future learning.

The author's experience in Computer Science Basics teaching is connected with teaching first year non-IT students at university, schoolteachers and school pupils of different ages. They all have one common feature – they are all beginners in computing. However, they all differ in their degree of motivation: university students are non-motivated or low-motivated group; schoolteachers are more motivated and school pupils are very motivated learners.

There is need to specify that Computer Science Basics content for these groups of learners also differs. University students have compulsory course in their curriculum, which consists of two parts: informative work and introduction to programming. Courses for schoolteachers and school pupils are non-mandatory and include only programming basics.

In order to apply the proposed theory into practice and to check its validity it has been decided to conduct an experiment with the Computer Science Basics course of the Department of Software Science at Tallinn University of Technology.

There was also an additional reason for this experiment - the course and approach to teaching needed revision. This course turned out to be rather difficult for most of the first year non-IT students. It resulted in lack of motivation and low examination grades. Moreover, as students' feedback showed in 2010, their comprehension of Computer Science Basics topics was at the critical level – 1,6 out of a possible 5. As follows from surveys of the first-year students, it was associated with a low level of school knowledge and, as a result, heavy perception of the subject.

Enumerated facts demonstrated that it was necessary to redesign the educational material and restructure the learning and teaching processes in the way that course content would be easy to understand but would still achieve the goals. The literature reviewed showed that more adaptive learning tools and taking into account individual properties of each student would motivate them and, as a result, would lead to better assimilation of new knowledge.

Moreover, it became necessary to develop an approach that should help beginners in Computer Science Basics and particularly in programming to overcome first difficulties in this field. In recent years, due to the rapid development of visual programming environments such as Scratch [11, 12], which is aimed just to beginners, it became possible to facilitate the transition to a coding.

Thereby, computing courses taught by the author began to serve as a material for the experiment and the area for applying the new approach to teaching and learning.

The central object of present research is the set of teaching strategies that could help first-year non-IT students to overcome difficulties related to the study of Computer Science Basics. These strategies are obtaining the information about learners' prior knowledge and their learning styles with their subsequent integration into the model of the student.

Research Questions

General research question of this thesis is "*How to help novices to overcome complexity of Computer Science Basics learning?"*

After literature analysis this question has been divided into two sub-questions:

The first is "*How to combine study about learners' level of readiness with theory of learning styles?"*

The second is "*How to use this combination in practice with the aim to individualize teaching and learning?"*

To answer these questions it is necessary to work out new teaching strategy and apply it in practice with the aim to validate the correctness of chosen approach.

Moreover, basis on the conducted research, the author attempts to give some recommendations on the use of appropriate forms, methods and organization of educational process.

The author has identified these questions based on the literature review, Computer Science Basics course teaching experience and main difficulties in teaching and learning processes. Moreover, main students' learning challenges, which have been identified during some years, were taken into account.

Methodology and Novelty

The author tackled named research questions by developing the teaching and learning model, which can help first year non-IT students to overcome the difficulties when studying Computer Science Basics course.

Regarding the methodology used, the author choice was *action research* because only during the teaching practice it is possible to identify and after to fix problems in teaching techniques used.

Experimental data for the research were collected from test results, learning styles questionnaire and university study information system. Test results were used to define students' level of readiness for new information perception and analysing their progress. Learning styles questionnaire was the basic for defining students' preferences.

Grades and feedback from study information system were used to follow students' progress and validity of teachers' work. Collected data were analysed quantitatively and qualitatively: calculations and observations.

Present research includes stages of the experiment with Computer Science Basics course content, its modernization, learning materials innovation and teaching model improving. The last one includes pedagogical and didactical techniques that should provide students with necessary conditions for successful study. Main strategy here is combined and applied information about learners' prior knowledge and their learning styles according to Richard M. Felder [13].

In addition, the author has developed new didactical strategy, which can help non-IT learners, which are usually beginners in coding, to grasp the main programming concepts. This methodology is based on the implementation of the visual programming elements before serious textual coding. Moreover, this visual programming environment and its elements are the basic tool for visualization in textual coding.

The novelty of the chosen approach is combination of teaching and pedagogical strategies with the aim to achieve maximum individualization in teaching and learning processes. These combined techniques are:

- e-support for students – students should have an opportunity to study efficiently not only in class;
- taking into account students' level of prior knowledge - *"what learners know before me"*;
- taking into account students' learning styles - *"how they learn with me"*.

This approach has been studied and has found its practical use in the Computer Science Basics course for the first-year non-IT students at the Department of Software Science at Tallinn University of Technology.

The practical novelty of this thesis lies in the teaching approach to programming courses for beginners. This approach has been developed based on global trends in teaching programming basics and focuses mostly on the model and algorithm building and their visualization, rather than teaching syntax and coding techniques. For novices the visual environment performs the role of the first programming language and afterwards acts as the tool for visualization and algorithm representation in further training.

Contributions of this thesis are:

- the new approach to Computer Science Basics teaching - the new combination of pedagogical strategies, which was successfully applied on practice;
    - this approach is designed to the first year non-IT university students with the aim to overcome main difficulties during the learning process;
- designed and applied technique for teaching programming for the novices;
    - this technique is designed to motivate non-IT students and to interest school pupils and schoolteachers in learning programming basics.

Organization of the Thesis

In *Chapter 1* of this thesis the author presents an overview about the state-of-the-art in the related fields of the problem investigated in the thesis. In addition, this chapter provides a short overview of the situation in the teaching of Computer Science on the international level and in Estonia. The author gives an overview of the Computer Science Basics course for first year non-IT students at Tallinn University of Technology with its outcomes and main difficulties.

Chapter 2 describes the Computer Science Basics course innovation, which consists of three stages named "e-course", "prior knowledge" and "learning style". The main idea here is students dividing into groups according to their backgrounds and learning preferences. In addition, this part of the thesis gives an overview of Richard M. Felder learning style theory, which was the basics of the third stage of the experiment.

In *Chapter 3* the author of the thesis statistically analyses the experimental data using Student's t-test for the comparison of the two means and describes original results of the research.

Chapter 4 provides readers with some didactical recommendations for teaching programming basics for the beginners and describes some active learning methods that can be applied during the programming studying. The main concept of this part of the thesis is an introduction to the basics of programming through a visual programming implementation to the Computer Science Basics course. Visual programming environment Scratch was chosen as the introductive tool for non-IT beginners. In addition, in this chapter of the thesis the author describes how it became possible to overcome the main difficulties in programming concepts learning and teaching to non-IT beginners. The main principles there is model-based programming teaching and the visualization of the algorithms.

Abbreviations

CS Computer Science

ECDL European Computer Driving License

IT Information Technology

PC Personal Computer

TUT Tallinn University of Technology

UML Unified Modelling Language

VBA Visual Basic for Applications

ÕIS Õppeinfosüsteem (Study Information System)

List of Figures

List of Tables

1 REVIEW OF THE STATE OF THE ART

This chapter provides a background information about Computer Science new trends and teaching. In addition, current chapter brings an overview of present situation in the level of knowledge in computing field of matriculants at Tallinn University of Technology. As well, this thesis reader can find general information about the TUT Computer Science Basics course for the first year non-IT specialities in this chapter.

1.1 The Overview

In recent years has greatly increased the interest in IT, particularly at schools. In this regard, several countries have carried out thorough investigations of the use of information technology and courses on Computer Science in different educational institutions. Analyses have shown that most of the courses do not meet the needs. As a result, several new curricula have been proposed to improve the situation.

1.1.1 Computer Science Trends on the International Level

In 2011 the new CS standard, "CSTA K-12 Computer Science Standards" was created in USA [13]. It sets out the basic requirements for the various areas and levels of the curricula. A number of courses and subject syllabuses were created on this basis. One of the most outstanding is the new CS syllabus “AP Computer Science Principles” [15] created under the support of US National Science Foundation. The work started in 2011 and the course is complete in 2016.

The documents mentioned above are based on the notion of "Computational Thinking", which defines general principles for describing the problems and solving them by means of software systems, including such concepts as abstraction and modelling, algorithms, data and information, programming, communicating and collaborating. A large part of the concepts is related to algorithms and programming [16].

In 2012 a comprehensive study "Shut down or restart?" was published by The Royal Society United Kingdom [17]. The research brought out significant shortcomings and offered ways to solve them. "Computer Science: A Curriculum for Schools" [18] was set up and published, where *Computational Thinking* is the main idea. Starting from September 2014 the course Computing (Computer Science + Information Technology + Digital Literacy) is included in the United Kingdom schools curricula [19].

1.1.2 Teaching Computer Science

What is the Best Way to Teach Computer Science to Beginners? Some discussions about the answer to this question can be founded in article [20]. The author of this thesis do agree with the statement that it is not possible to teach programming

using PowerPoint, students need a visualization of their code. Undoubtedly, described presentation platform is useful tool. However, what about the novices who do not have any programming experience? Do not they need to see something simpler to start with?

The teaching of programming concepts and algorithms forms a fundamental part of any Computer Science and Engineering degree [21]. Moreover, Computer Science should be taught to non-IT students with the aim to develop their algorithmic thinking and general skills. There are many differing ideas about what exactly should be taught to them. The main point of disagreement there is programming - should it be included in curricula for non-IT students or not and what is the proportion of it.

Computer Science includes a quantity of topics and has a quantity of definitions [22, 23]. However, it should be clear what Computer Science means for non-IT students and what kind of knowledge it brings. The main issue for educators here is to create a curriculum, which meets the needs of students regardless of their specialty. Knowledge, which students obtain should be applicable and have the potential for further development.

With regard to students, it should be noted that the majority of them always face difficulties throughout the learning process. This implies another challenge for teachers – to help students to overcome these difficulties using modern techniques and methods in cooperation with fundamental educational principles.

The choice of topics for the Computer Science course is greatly simplified by university requirements and restrictions. Moreover, there are obligatory computer skills and knowledge, without which it is impossible to work with the computer.

Curricula are composed. The next milestone is to attract non-IT student' interest to any IT subject. On this stage the teacher requires not only his knowledge in this area, but also pedagogical skills and the ability to apply them in practice.

Literature review shows that there are plenty of attempts to reform Computer Sciences courses and all of them lead to success in selected area. Teachers and scientists working on the development of Computer Science and on the best way to teach it.

Doctoral thesis [24] analyses the challenges of the instructional process at a university of technology from the viewpoints of students, teachers and the university administration. One of the raised questions concerns the difficulties students encounter when they study computer programming. The question is examined and thoroughly studied. However, in present thesis author's opinion, such aspect as the individualization of teaching and learning is not taken into

account here. Students are considered in general with their general difficulties. It can be assumed that students' difficulties in the introductory programming course arose not only from the mentioned reasons such as lack of time or motivation to study, but also from the fact that the everyone's learning manner is quite different.

Several challenges and ways of their solutions this thesis reader can find in article [25]. Some principles and methods of active learning in Computer Science and effective using of knowledge about students' preferences can be founded there. Non-motivated students - this is a common problem of this study and present doctoral work. Here it should be noted that the introduction to programming could begin with something attractive to understand the basics, for example with Scratch. This tool raises students' motivation and performs the role of a program action visualizer.

Computer Science courses for business computing students were considered in article [26]. The research participants made 10 causal attributions that were either cultural or specific to computer programming. The study outcomes can be used to suggest ways to energize unmotivated students. Main research method there is interview method; however, the author believes that anonymous surveys provide much more information. In addition, it is possible to learn more if students are not aware of an ongoing experiment or research.

Basic practices in education reform on software programming courses for non-IT-majoring undergraduates readers can find in [27]. Main concerns there are textbook revising, teaching model reforming, student demand orientating, knowledge transferring, teaching procedure designing and teamwork conducting. However, the aspects of the student's personality are not taken into account. How do they perceive the learning material transferred to them? Is textbook revising enough for successful assimilation of knowledge?

In study [28] was founded that students with no experience in programming languages, is actually quite amenable to relating well to the principles of programming in logic as long as those principles were presented them in an adequate way. Educators motivated them through presenting applications of their interest, focusing on possible interdisciplinary uses. The author of this work is totally agree with active learning theory, however using it teacher have to know who students in his class are. For some kind of students this kind of work does not suit by reason of their individual learning style.

The paper [29] presents an approach of teaching computer programming courses to first year students using agile principles and practices of pair programming [30, 31]. It intended to increase the chances of engaging first year students and assist the teachers to reflect on their teaching as well as their students' learning. It is ideal situation when students and teacher are partners in the learning process, but is it always possible, especially when we are talking about first-year non-motivated students whose prior knowledge is at zero level?

Considered sources show, Computer Science is quite complicated for all kind of learners, especially for non-IT and novices in this field. These difficulties arise due to various reasons: complexity of learning materials, types of learning materials, lack of knowledge base, lack of supporting materials and etcetera. At the same time, it should be noted that there are variety of strategies, which works successfully and greatly helps learners to overcome these challenges. However, practice and literature show that problems in this field remain and they must be solved using pedagogical and didactical techniques in their combinations with Computer Science.

Thus, it can be concluded that if described teaching strategies work, then they should work even better and more effectively and bring better results if they are merged successfully.

It should be mentioned there that the course, which is considered in current thesis as a model of research is not only programming course and its innovation and modernization is the deep work with all the components of the educational process: learners, teachers, methodologies, materials and content.

1.1.3 Level of Knowledge of Matriculants at Tallinn University of Technology (non-IT Specialities)

Annually, in the beginning of the programming course for school pupils at TUT, the author of the thesis carries out short surveys about computer science at Estonian schools. Their results show that the current situation in teaching and learning computing is quite discrepant. According received responses some schools do not have Computer Science lessons at all or this is an elective subject with small number of pupils. In some schools it is taught only for two or three years, which is a very short period to prepare students for the next, university level.

This drawback is associated with two main reasons. The first one being that there is no nationwide computer science curriculum in Estonia. The second one is that Computer Science subjects are not mandatory in Estonian schools.

A logical consequence of these reasons is the situation where each schoolteacher introduces learners to the material at his own discretion: certain pupils draw in Paint or in SketchUp, others learn the computer hardware in theory, etc. In connection with this, the level of PC skills among non-IT learners falls every year and reduces to commonplace Facebook usage or playing games.

The following diagram shows the last years statistics about non-IT students testing at the beginning of the Computer Science Basics course (Fig. 1.1). The test content is based on the topics described in the European Computer Driving License (ECDL) [32]. Test questions and tasks focused on creating documents and

presentations, processing spreadsheets, and elementary knowledge in programming.

The feasible maximum number of points is 100. As can be seen, the level of computer skills is quite low and steadily decreasing. Moreover, during the last year of the research it fell sharply by more than 19 points.

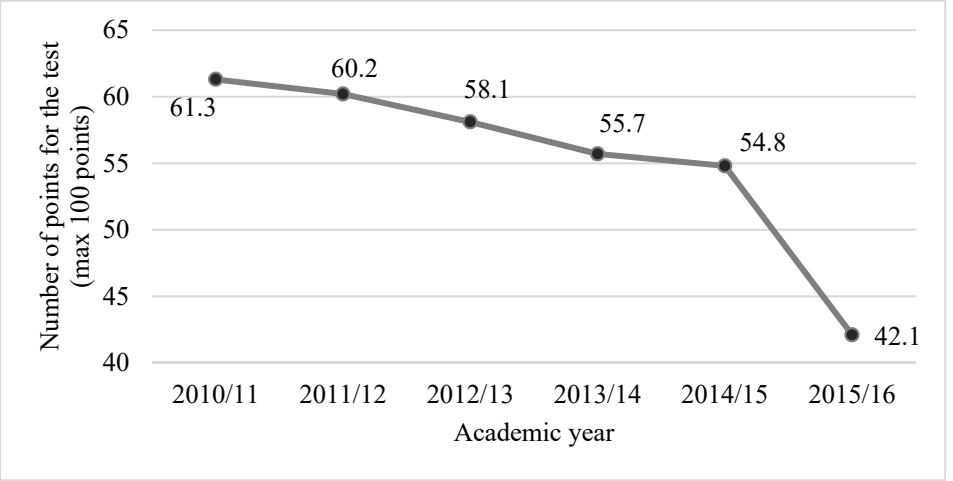

Figure 1.1 The averages of the beginning test

Students' feedback about the test results shows that they do not have enough prior knowledge for TUT Computer Science Basics course for three main reasons:

- at school they did not interested in computing;
- their skills were enough for daily studies at school (small documents and presentation);
- at school they were not taught these topics.

In the present Computer Science Basics course the author has to consider mentioned facts and build the curricula accordingly.

It should be mentioned there that recently programming, robotics and so forth courses for school pupils are carried out throughout in Estonia. Such courses are held at Tallinn University of Technology to and author of the thesis takes part there as a programming teacher of the course named 'Basics of Applications Development and Programming'. These courses are very popular among learners from 10 to 18 years who are interested in IT (*Chapter 4*).

Nevertheless, pupils who have obtained sufficient informatics-knowledge at their schools or additional courses often become IT students at university level. Unfortunately, they are not current Computer Science Basics course audience.

It should be noted there that several countries research shows decrease in the level of the computer skills in schools. Especially deep research has been performed in US CSTA [33] and UK [34].

1.2 Computer Science Basics course for non-IT at Tallinn University of Technology

1.2.1 The Course Description

The Computer Science Basics course belongs to the curriculum of the Department of Software Science at Tallinn University of Technology its name is Informatics [35].

The aim of the Computer Science Basics course, designed for the first year non-IT students, is creation of applications by using standard PC equipment and developing object-oriented computational thinking. In this context the term *Computational thinking* is can be interpreted as looking at a problem in a way that a computer can help person to solve it.

The International Society for Technology in Education and the Computer Science Teachers Association defines *Computational thinking* as:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them;
- Logically organizing and analysing data;
- Representing data through abstractions such as models and simulations;
- Automating solutions through algorithmic thinking (a series of ordered steps);
- Identifying, analysing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources;
- Generalizing and transferring this problem solving process to a wide variety of problems [36].

Computational thinking is the thought processes involved in problem-solving so that the solutions are represented in a form that can be effectively carried out by an information-processing agent [37].

Building the Computer Science Basics course, the author of the thesis and the author's colleagues try to follow mentioned *Computational thinking* definitions and compose learning and teaching materials accordingly.

The course lasts two semesters and number of weekly study hours is two or three. Group size is usually 20-30 students.

The learning process starts with processing information and creating applications using MS Excel spreadsheets: formulas, diagrams, built-in functions, tables, conditions, sorting and search, requests, diagrams and other facilities. Moreover, students learn the general principles of the different types of the documents creation and processing using MS Word and MS PowerPoint. In addition, Google tools as an alternative to the Microsoft products are used in the teaching and learning process. Model-based and object-oriented approaches are applied during the study.

Further, during the course second part, students learn the basics of programming in practice and the main principles of modelling and algorithmization. Python and Visual Basic for Applications (VBA) have been picked out as the programming languages for the second section of this course.

It should be noted that the programming part of the course is usually sufficiently complicated for most of the non-IT students, especially for the humanitarians. This issue is solved by implementing visual programming elements in the course curriculum and helps students to take on board the main programming concepts (*Chapter 4*). Positive results of the named strategy is demonstrated in [38, 39].

The set of practical assignments depends on the students' specialization: economics, social, chemistry and civil engineering.

During the course instructors apply classic face-to-face classroom methods, group works and independent learning in Moodle e-environment [40].

The author and author's colleagues have chosen Moodle for learning and teaching for three reasons:

- Moodle is increasingly used in schools and universities;
- Moodle is used in Tallinn University of Technology;
- Moodle continuously develops.

The last one gives teachers a huge amount of different opportunities for individualizing the learning process, such as adjustment of the learning pace, for example, as well as increase and variety in the number of learning assignments. Furthermore, students get a diversity of ways to learn and possibilities for self-tests in the e-part of the Computer Science Basics course. During the study time, they can choose between different kinds of teaching materials and use what they prefer based on their knowledge and learning styles.

1.2.2 The Course Outcomes

In the Computer Science Basics course its authors adhere to principles outlined in Computer Science Curricula 2013, such as:

- Computer science curricula should be designed to provide students with the flexibility to work across many disciplines.
- Computer science curricula should be designed to prepare graduates for a variety of professions, attracting the full range of talent to the field.
- Computer science curricula should be designed to prepare graduates to succeed in a rapidly changing field [41].

Named principles are especially important for the Computer Science Basics course building and developing because its audience are first-year non-IT students who have to apply their IT-knowledge during their study at university and in future work.

The main learning outcomes that are documented in ÕIS [42] in the Computer Science Basics course are listed below. ÕIS is e-environment, where students and teachers get information about the courses and curricula, students declare courses, keep results and give anonymous feedback on their educational process. Detailed courses descriptions this thesis reader can find in Appendix.

Student who complete the course:

- acquires independent work skills bases;
- logically and argumentatively explains selected tools and methods feasibility for solving the task - has the computational thinking skills
- knows and applies general principles, methods and tools of the documents creation;
- is familiar with the general principles of creating applications, methods and tools;
- 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 using parameters and arguments.

The purpose of this study is to demonstrate a teaching approach and some teaching strategies in the Computer Science Basics course for the first year non-IT students at the Department of Software Science. The author suggests some solutions for making the course, which is usually complicated, more dynamic and

attractive, thereby raising students' interest and keeping their motivation with the aim of achieving the learning outcomes set.

1.3 Conclusion and discussion

Based on the foregoing it should be noted that most of above-mentioned sources consider difficulties, which is related to teaching Computer Science Basics and programming to novices and techniques to solve them.

However, the author should note that some aspects have not been taken into account in considered researches.

First, it is necessary to understand the differences between students in a classroom and their learning styles. It is necessary to build the educational process proceeding from this knowledge.

Secondly, before submitting educational material to students, it is useful to analyse it from the point of view of any student. What is more, it is necessary to consider that pupils' level of prior knowledge can be very different. On this stage questions like *How will a student perceive this material? Does learner have enough prior knowledge to understand this? Should student teach something else for this?* - are very useful and help to analyse the educational material and make it better.

Thirdly, for novices it should be clear from what and how they start to write their first program. Maybe their first outcome should not be a written program but be only a model, assembled from available parts? This starting level should be simple and understandable for all of them.

Fourthly, is it possible to visualize programs execution for making it more clearly without any additional tools on the basic level?

Fifthly, in programming teaching to non-IT students, it is necessary to determine the required amount of knowledge. Should students know any particular programming language or it is enough for them to have skills to build an algorithm for the problem solving and be able to explain it to specialists?

Finally, it is necessary to motivate non-motivated students and maintain high level of interest among all learners.

Concerning Computer Science Basics course at Department of Software Science of Tallinn University of Technology the author should mark that during present research this course has been entirely updated. This reconstruction starts with the renovation of educational material and its presentation to students and continues with the development of the system of the teaching programming for beginners. During this process the author and author's colleagues proceed from the actual needs of today's learners and their learning preferences.

Furthermore, proceeding from the fact that this course is designed for the first year university students, Computer Science Basics at schools have to be certainly considered during these updates. It is necessary to take into account first year students' prior knowledge to provide them with new, quality and contemporary knowledge.

In the current study for innovating the Computer Science Basics course and individualization of the learning process the author can name the main strategies – students dividing into groups according to their backgrounds and a maximally effective use of e-environment with its possibilities.

In the conclusion, it is necessary to note that people learn in different ways and all kind of learners should be provided with the best conditions to acquire knowledge.

Having studied literature the author of this thesis has chosen three main directions in research, has combined them and has gradually applied them in practice. These directions are:

- continuous e-support for students during their learning process;
- taking into account students' level of prior knowledge;
- taking into account students' learning styles.

Using above-mentioned combined strategies the author intends to help non-IT students to overcome main difficulties in studying Computer Science Basics.

In the next chapter of this thesis the author presents stages of the experiment with the Computer Science Basics course modification and gives an overview of the basics of the chosen methodology. New teaching strategy has been created based on this experiment.

2 THE COMPUTER SCIENCE BASICS COURSE INNOVATION

In the current chapter of this thesis the author presents the Computer Science Basics course modification and innovation experiment stages and gives an overview of the basics of the chosen methodology – student division according to their backgrounds and preferences. The basis for the division according to preferences is Richard M. Felder's learning styles theory [43]. Under the background the author of this thesis understands a level of prior knowledge of each student.

Thus, the author has succeeded to combine the learning style theory and prior knowledge level. This combination has allowed to individualize learning even more deeper by considering each student as a set of certain parameters. These parameters has gave instructors the opportunity to compile teaching materials, which are necessary and useful exactly for each student.

This chapter is based on the paper A (*Mironova, O.; Rüütmann, T.; Amitan, I.; Vilipõld, J.; Saar, M. Computer Science E-Courses for Students with Different Learning Styles*).

2.1 The Background

The main keynote of university is continuous communication and close collaboration between pupils, educators and scientists. Unfortunately, university instructors cannot provide all students with one-to-one tutoring. However, this fact has not affected the main aim of the educational process – to guarantee high-quality knowledge and modern skills. The teachers work there is the art – the art to present and transfer knowledge and skills regardless of the circumstances.

During the experiments with the Computer Science Basics course structure and content, were considered the differences in students' characteristics, especially their background (the faculty and the level of prior knowledge) and their preferred learning styles. The question remained how to achieve as much individualization of teaching as possible, using the existing time and personnel resources.

Since 2010, a group of lecturers started applying a new approach to the design of the Computer Science Basics courses for economics, social, chemistry and civil engineering faculties. Considering the fact that students is the centre of learning and they actively constructing this process [44], innovations should be targeted to them. Year by year this approach has become more flexible and adaptive to the nature and preferences of every student.

At the beginning, educators, who were involved in the experiment, have randomly divided all the students into equal *reference* and *test* groups. The

division is not linked to the students' specialization. The average number of members in each group is about 150; it varies depending on the annual general number of students at Tallinn University of Technology. It should be noted that students are not aware of the research.

The *reference* group is taught using the same course materials but these students are not supported with any additional systems. The students of the *test* group are directed in choosing their e-learning materials based on the data obtained through the tests in the e-environment.

The intention here is to compare the results of these two groups at the beginning and at the end of the course.

At the beginning of the course the students are tested to find out their level of knowledge in the Computer Science Basics. Experience has shown that such testing is necessary for the development of the course content. The purpose is to keep track with new times and main trends, as the Computer Science is one of the fastest-developing sciences [45, 46].

The nature of the tests for both groups is similar and based on the concepts defined in the European Computer Driving License. The assignments focus on some principles of the work with the PC, like creating text documents and presentations, handling information using the spreadsheets, and elementary programming knowledge. Tests include both practical and theoretical tasks.

The programming category of the questions was added to the test some years ago and is currently it develops rapidly. The author bears in mind that a new elective course "Basics of Applications Development and Programming" was recently included in the secondary school curriculum in Estonia (*Chapter 4*). Some of these materials are also used at Tallinn University of Technology for teaching programming basics for first year non-IT students.

2.2 E-course

Higher education institutions are taking opportunities offered by the eLearning community to design and offer educational environments that accommodate various learners' educational needs [47]. E-learning allows modern learners with a personal computer, connected to the internet to attend the course anywhere and at any time. It is very important for students to participate in the educational process, to get the material or submit the homework, on time regardless of whether the person is or is not at the university.

The first phase of the Computer Science Basics course innovation, which was named "*e-course*", includes the adaptation of educational materials for Moodle e-environment.

Uploading a set of lecture materials and exercises into a learning environment does not ensure that students comprehend it and obtain necessary

knowledge. Therefore, in order to make study materials suitable for an e-course all teaching materials (theory as well as materials for practice) were thoroughly revised. The aim was to provide an effective delivery of the online content. To achieve this goal was aimed at working out a new pedagogical and didactic policy as well as strategies for the new e-course.

Theoretical materials were innovated and supported first of all with learning videos. It should be noted that in the Computer Science Basics course is used not only video lectures but also short screen-captures, which explain the most complicated tasks. Creating these videos, the course teachers adhered to the principles of Khan Academy [46].

Practical tasks of the course were reconsidered and supplemented with various group and pair work tasks and self-tests. With regard to self-tests it should be noted that learners are afraid to make mistakes especially when they doing something for evaluation. Because of this fear, they do not analyse their mistakes and make them repeatedly. Self-tests give students the opportunity to solve tasks calmly focusing on content, not on grade. During this educational work learners can much more think and analyse it.

These innovations made the Computer Science Basics course more dynamic and attractive for the first year non-IT students. Both groups, the *test* and the *reference* group got access to this renewed course.

At the same time, educators have not abandoned the standard lessons in computer classes because e-learning cannot completely replace live dialogue between learner and teacher. Face-to-face lessons were held as usual but now instructors got many advantages. Due to the e-lectures and visual explanations in Moodle they had more time for practical training in contact lessons. It is necessary to mention that students have no access to practical exercises unless they solve the tests, which are based on the theoretical material of each topic. Thereby students come to the lessons already sufficiently prepared for the practical tasks. Often they get a few small practical tasks in the e-environment and afterwards, in class, they use already ready solutions to solve the bigger tasks.

It is generally known that effective computing is impossible without practice. During face-to-face lessons students work in pairs or groups gives an opportunity to try the obtained knowledge in practice. Moreover, such kind of work develops teamwork skills, which is very important, especially for the first year students during their first semester.

In such case, the role of the classical educator is slightly different – the lecturer becomes more of a supporter in the students' teamwork of their learning assignments.

During contact lessons learners have an opportunity to ask questions related to their homework and share their practical skills and experience with the

rest of the group. Practical knowledge transferred in such a way is obtained much faster than in standard lectures. This fact was confirmed by the results of students' feedback retrieved from ÕIS (Fig. 2.1).

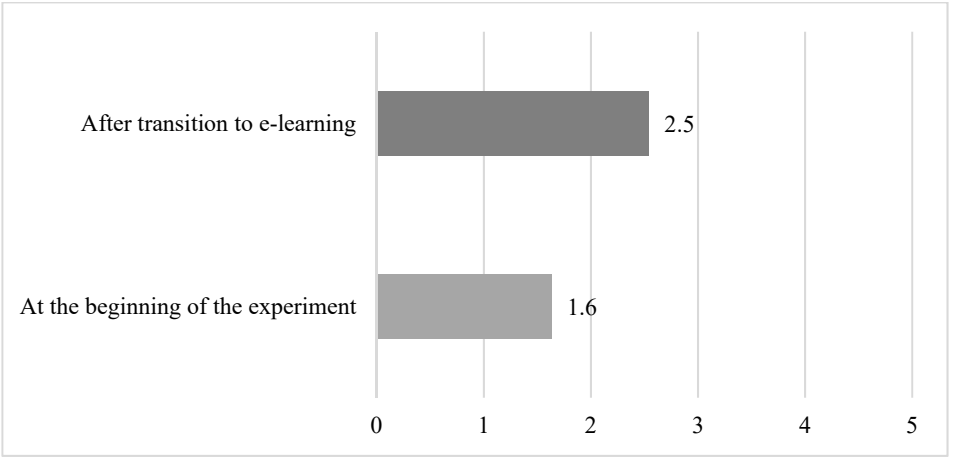

Figure 2.1 Computer Science Basics topics comprehension after the first step of the experiment

In their feedback students named another advantage of such practical lessons: they have an opportunity to get support or to ask something not only from their teacher (sometimes they hesitate to do it) but from other students, too. It should be mentioned that this form of support is equally important and useful for both sides: the one who gets it and, especially, for the one who gives it. To detect, explain and, afterwards, to correct a mistake in calculations or in the program code is a substantial practical skill in Computer Science subjects.

In addition, in an e-environment students get their practical assignments in accordance with their specialities. However, these assignments are still united under a common subject topic. For example, students from the economics department get more tasks related to table calculations; social sciences students implement the information filtration, statistics calculations and various requests.

It should be noted that the course materials are organized sequentially and it is not possible to get a new portion of theoretical materials and practical tasks without solving the previous ones. Thanks to checking opportunities in the e-environment, like tests or self-tests, teachers do not have to spend time on routine inspection of the assignments at all. Using the automated checking tools gives students an opportunity to learn within their own pace. They do not have to wait for the feedback on the assignments from the lecturer and his/her manual permission to proceed onto the next level. E-checking systems do it faster and as many times, as is needed.

In addition to the aspects mentioned above it should be noted that now, in the renewed course, teachers and learners started to use e-course forums very actively. This way the students get an opportunity for online communication and fast online help. The aim was to show that they can get support and advice any time, and can share their ideas, problems and solutions. Through these forums the course instructors often get perfect brainchildren for group work and individual assignments.

Another advantage that non-IT students have when they participate in the e-course is that their technical skills improve.

Transition to Moodle e-environment gave educators the opportunity to follow students' progress and it became fairly easy to get the statistical data of different samples for analysis, development and improvement of the implemented learning methods.

It makes no sense now to recount all the advantages that the author achieved during the first stage of the modification of the Computer Science Basics course. Such benefits have already been systematized and described in detail [49]. However, the first positive results of the work: students' feedback and increase in academic achievements. The author does not present specific data in numbers in this work because the transition of the course into e-learning took place a long time ago.

For confirmation the correctness of the chosen direction here should be mentioned the study, which provides evidence that a well facilitated e-learning setting can indeed enhance learning motivation and student efficacy [50]. In addition, the results from this research have provided insights to educators who are keen on using technology in their teaching.

2.3 Prior Knowledge

Starting from the second stage, which the author entitles "*prior knowledge*", experimental work takes place only with students from the *test* group. The students of the *reference* group were taught as usual.

The impetus for this step of the experiment was the fact that the majority of students do not have enough knowledge acquired in school to study the Computer Science Basics course at TUT.

At the beginning of the course educators start with dividing the students into three e-streams based on their readiness for Computer Science Basics course. This division was realized through an e-test and implemented in Moodle e-environment. The students, however, were not aware of the experiment.

Questions of the test have different level of complexity and different "price" (in points). Test questions content is based on the knowledge, which are necessary to start the TUT Computer Science Basics course.

In the described division it was proceeded from the level of students' knowledge required to start the course. Those students whose e-test results were more than 85 points was named "*experts*", "*advanced users'*" result was between 60 and 85 points, other students were called "*beginners*".

The named groups of students receive different amounts of practical and theoretical tasks in Moodle, with different levels of difficulty. To move to the next topic the mandatory set of exercises has to be solved. In the e-environment, the "*beginners*" have to solve their set of tasks and "*advanced users'*" tasks before they get access to "*experts'*" exercises – the main material of the course curriculum. "*Advanced users*" have to solve their tasks and then can proceed to main topics.

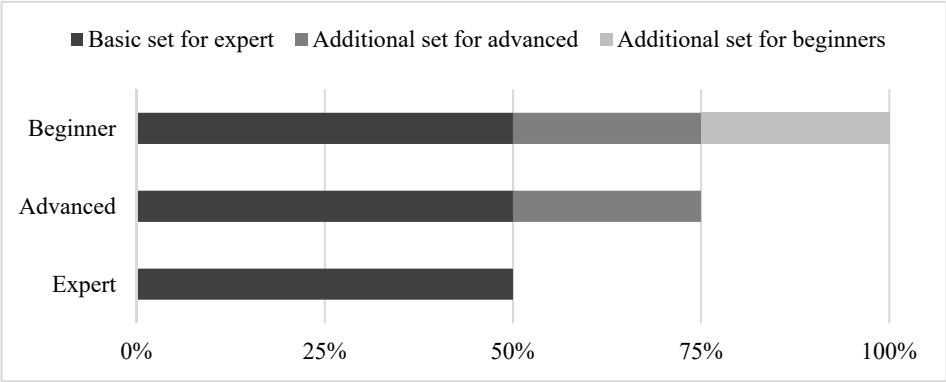

Figure 2.2 The distribution of the set of practical tasks and tests

These additional sets of tasks are catered for exactly what learners need to know for the current Computer Science Basics course. Students do not need to pay for any additional IT-courses anymore and they get all the materials and assignments centrally, in one place – Moodle e-course, in parallel with their main studies.

To automate and speed up the checking of the increased number of tasks a special e-tests system was developed.

Thereby, it was possible to increase the amount of practical assignments for students with different levels of readiness without increasing the subject hours and students' load. This stage of the course innovation gave the course teachers an appropriate level of the students' readiness for face-to-face lessons.

The above mentioned method provided the Computer Science Basics course teachers with actual and important information about what the learners knew before starting the course – their school level of knowledge. Every year educators get an overview of the current situation of Computer Science subjects

at secondary schools in our country. Moreover, according to the results, they are able to provide students with all necessary learning materials.

Students' feedback again shows the raise of the level of the subject understanding that confirms the properly of the chosen research strategy (Fig. 2.3)

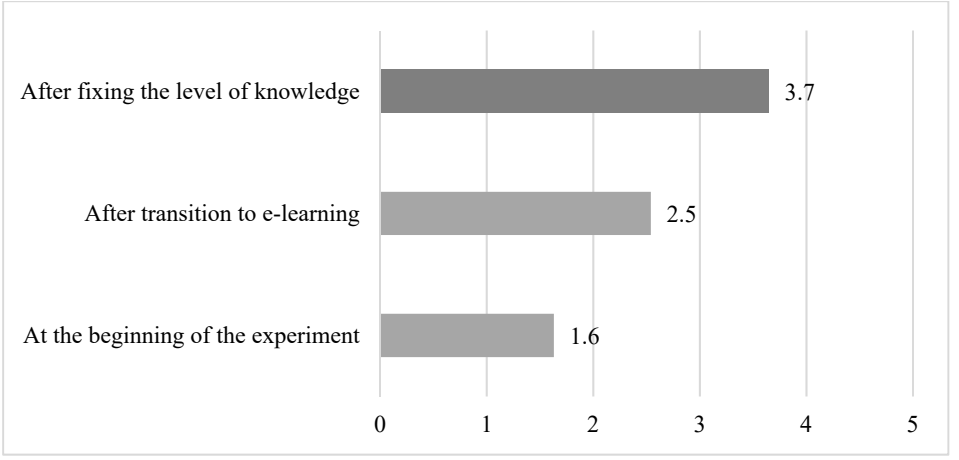

Figure 2.3 Computer Science Basics topics comprehension after the second step of the experiment

2.4 Learning Style

The third part of the innovation experiment with the course modification is the "*learning style*" phase and it was also realised in Moodle e-environment. This stage is a continuation of the *test* group's students division into groups. In addition, it was possible to maximize the use of Moodle e-opportunities.

Major work in the chosen direction is studying and analysis of students' data and adaptation of the curricula [51] and teaching materials in accordance with their interests, goals, preferences and individual characteristics.

The term "learning styles" means the understanding and acceptance that every person learns differently. For learners this is preferential way, in which they absorb, process, grasp and retain an information.

Learner's preferences and requirements play an important role in educational process. In the literature, there are a variety of learning style models. These deep researches about the individualization of learning depending on students' characteristics [52 - 55]. They all have similar features based on psychology and physiology. Moreover, a great deal of learning strategies have worked out with the aim to help learners to study better. In addition, adaptive and intelligent web-based educational systems should be mentioned in source [56].These systems build a learner' preferences, goals and knowledge model and

after use it during the interaction with the learner. Named interaction aim is to adapt to the needs of that learner. With the knowledge of different styles, the system can offer valuable advice and instructions to students and teachers to optimise students' learning process [57].For example the research [58] focuses on the consideration of learning styles and cognitive traits in adaptive web-based educational systems. Authors investigate the benefits of incorporating learning styles and cognitive traits in these systems.

Short review of the different learner's models with references this thesis reader can find in [59].

In the current Computer Science Basics course Felder-Silverman model [60], was picked out as the basis for the distribution. According to this model developed by, a student's learning style may be defined by the answers to four questions:

1. What type of information does the student preferentially perceive: sensory or intuitive?
2. What type of sensory information is most effectively perceived: visual or verbal?
3. How does the student prefer to process information: actively or reflectively?
4. How does the student characteristically progress toward understanding: sequentially or globally [61]?

This model has caused the author's interest during the study and finally was chosen because it was constructed by experiences in engineering pedagogy.

There should be noted Richard M. Felder's merits in the field of engineering pedagogy: Global Award for Excellence in Engineering Education, International Federation of Engineering Education Societies (2010) and Lifetime Achievement Award in Engineering Education, American Society for Engineering Education (2012).

Similar study outcomes readers can find in [62]. Authors aim to introduce a personalized e-learning system based on the concepts of learning styles and particularly the Felder and Silverman Learning Style Model. In research [63] authors describe a personalized e-learning system, which can automatically adapt to the interests, habits and knowledge levels of learners. The differences between the learners are determined according to their previous knowledge of the matter, their learning style, their learning characteristics, preferences and goals. Theoretical justification of different aspects of personalization a framework for personalized e-learning development can be founded in [64]. The same authors propose ideas allows to adapt knowledge management principles to improve personalized e-learning processes and look at the topic of personalization of e-

learning even more widely [65]. Research [66] introduces an idea towards developing a personalized learning system based on automatic approach.

2.4.1 Learners' Types According to Richard M. Felder

Learning styles are characteristic cognitive, affective, and psychological behaviours that serve as relatively stable indicators of how learners perceive, interact with, and respond to the learning environment [67]. Students learn best when instruction and learning context match their learning style.

Depending on their learning style, Felder differentiates between the following groups of learners [68]:

- active and reflective;
- sensing and intuitive;
- inductive and deductive;
- visual and verbal;
- sequential and global.

For better comprehension of the topic the author gives a brief description of the learners' types in according with Richard Felder's theory [69].

2.4.1.1 Active and Reflective Learners

An *active learner* is a person who feels comfortable at active experimentation than reflective observation. Active learners acquire new knowledge best by doing, discussing and explaining it to others; they work well in groups. Active learners do not learn much in situations that require them to be passive. They tend to be experimentalists.

Reflective learners do not learn much in situations that provide no opportunity to think about the information being presented. They think, learn and work better by themselves or with at most one other person. Reflective learners tend to be theoreticians.

2.4.1.2 Sensing and Intuitive Learners

Sensing learners like learning facts, data, and experimentation. They like solving problems by standard well-known methods and dislike "surprises". Sensors are patient with detail but do not like complications. These learners are good at memorizing facts. They are careful but may be slow. Very often this slowness puts them at a disadvantage in time. This manifests itself in timed tests: sensors may have to read questions several times before beginning to answer them, they frequently run out of time.

Intuitive learners prefer principles and theories. They like innovation and dislike repetition. Intuitors are bored by detail and welcome complications. They are good at grasping new concepts, quick but may be careless. Intuitive learners

are more comfortable with symbols than are sensors. Intuitors may also do poorly on timed tests but there takes place a different reason: the impatience with details. Intuitive learners may start answering questions before they have read them thoroughly and to make careless mistakes.

2.4.1.3 Inductive and Deductive Learners

It is generally known induction is a reasoning progression that proceeds from particulars (observations, measurements, data) to generalities (governing rules, laws, theories). Deduction proceeds in the opposite direction. In induction one infers principles; in deduction one deduces consequences.

Inductive learners prefer to learn a body of material by seeing specific cases first (observations, experimental results, numerical examples) and working up to governing principles and theories by inference These learners need motivation for learning. They need to see the phenomena before they can understand and appreciate the underlying theory.

Deductive learners prefer to begin with general principles and to deduce consequences and applications.

An effective way to reach both groups is to follow the scientific method in classroom presentations: first induction, then deduction.

2.4.1.4 Visual and Verbal Learners

Visual learners remember best what they see: pictures, diagrams, flow charts, time lines, films, demonstrations. If something is simply said to them, they will probably forget it.

Verbal learners remember much of what they hear and more of what they hear and then say. They get a lot out of discussion, prefer verbal explanation to visual demonstration, and learn effectively by explaining things to others.

2.4.1.5 Sequential and Global Learners

Sequential learners follow linear reasoning processes when solving problems. These persons can work with material when they understand it partially or superficially. Sequential learners may be strong in convergent thinking and analysis. They learn best when material is presented in a steady progression of complexity and difficulty.

Global learners make intuitive leaps and may be unable to explain how they came up with solutions. They may have great difficulty working with partially or superficially understood material. Global learners may be good at divergent thinking and synthesis. These learners sometimes do better by jumping directly to more complex and difficult material.

2.4.2 Teaching Techniques to Address All Learning Styles

Based on the above definitions of learning styles Richard Felder has formulated some teaching techniques that should fit all of them.

- Motivate learning.
- Provide a balance of concrete information and abstract concepts.
- Balance material that emphasizes practical problem-solving methods with material that emphasizes fundamental understanding.
- Provide explicit illustrations of intuitive patterns and sensing patterns and encourage all students to exercise both patterns.
- Follow the scientific method in presenting theoretical material.
- Use pictures, schematics, graphs, and simple sketches liberally before, during, and after the presentation of verbal material. Show films. Provide demonstrations, hands-on, if possible.
- Use computer-assisted instruction.
- Do not fill every minute of class time lecturing and writing on the board.
- Provide opportunities for students to do something active besides transcribing notes.
- Assign some drill exercises to provide practice in the basic methods being taught but do not overdo them. Also provide some open-ended problems and exercises that call for analysis and synthesis.
- Give students the option of cooperating on homework assignments to the greatest possible extent.
- Applaud creative solutions, even incorrect ones.
- Talk to students about learning styles.

2.4.3 Educational Process in Accordance with Students' Preferences

Through a test, the author learned that the majority of the course participants in the test group were *active* and *visual learners* and they had very strong preferences for their learning process. These preferences were detected according to the Felder's test [70], which was held at the beginning of the Computer Science Basics course (in the fall semester). As shows Figure 2.4, each year the number of active and, especially, visual students increases (Table. 2.4).

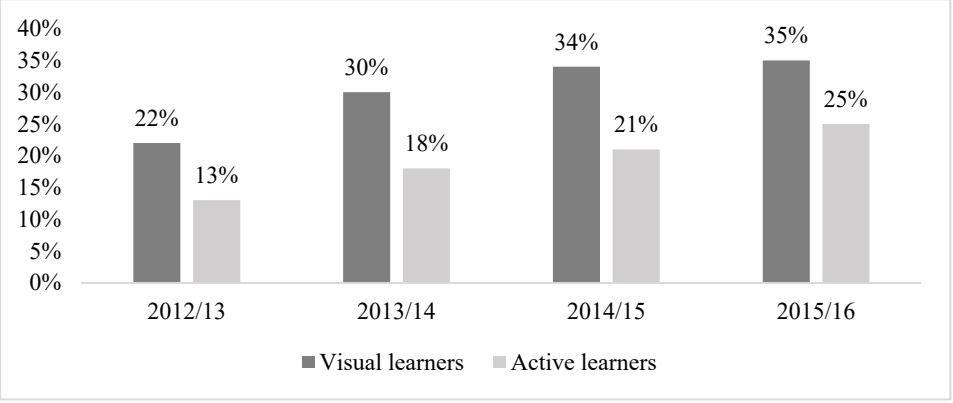

Figure 2.4 The increase of the number of the visual and active learners in the test group

Table 2.1 The increase of the number of the visual and active learners in test group

	2012/13	2013/14	2014/15	2015/16
Total	**78**	**89**	**75**	**84**
Visual learners	17	27	26	29
Active learners	10	16	16	21

Throughout the educational process, students were provided with necessary learning materials and activities in accordance with Felder's instructions [71].

For students' motivation different kind of assignments are used in the course: exercises based on other courses; projects, which are based on learners' experience and related to their work (many students work in parallel with their studies). Such assignments are very useful especially for inductive and global learners.

Active learners automatically received more group work and group homework assignments. Besides they got opportunities to help others – active learners could check and correct other students' works and assignments in Moodle (of course, a teacher controls this results and grades). The excellent guide to the effective design and management of students' team is the research [72] with its references.

Moreover, active students answered questions in the e-course forums and took the role of a tutor in face-to-face classes. It should be mentioned that they did it with pleasure and thereby their level of motivation constantly increased.

For visual learners, a great variety of visual representation of the educational materials (that was already mentioned above) was provided: video lessons, screenshots, and short video fragments of practical assignments.

Sensing learners got exercises, which were connected with solving real problems associated with other subjects or related to life situations.

Interests and preferences of the other types of learners were also taken into account: balanced educational material, intervals in class time to give students an opportunity to think and discuss about what they have been told etcetera.

The results of the experiment showed a positive trend in the acquisition of knowledge. Students of the *test* group managed with all practical assignments much better than students from the *reference* group. They did it with a great desire and showed more initiative in their work.

As shown in Figure 2.5, during the experiment examination results of the *test* group students were better. Numerical results are shown in the Table 2.2.

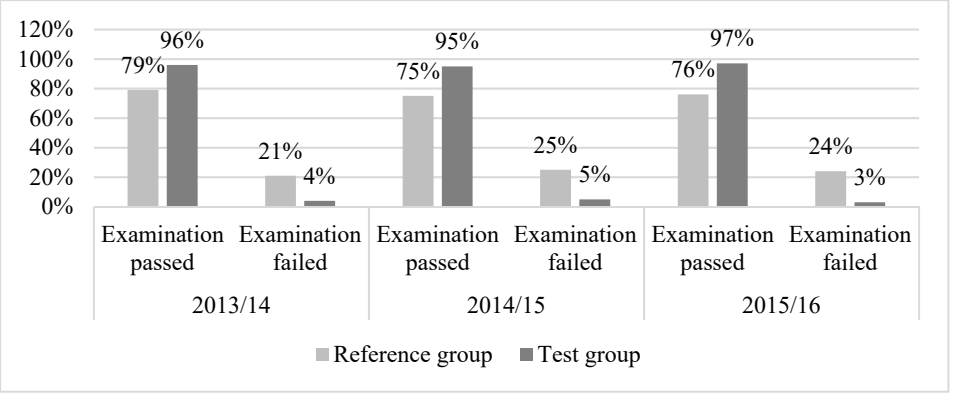

Figure 2.5 Examination results (%) of both students' group

Table 2.2 Numerical results of both students' group

	2013/14		2014/15		2015/16	
	Reference	Test	Reference	Test	Reference	Test
Examination passed	70	85	56	71	64	81
Examination failed	19	4	19	4	20	3
Total	**89**	**89**	**75**	**75**	**84**	**84**

Finally, students' feedback showed that course topics comprehension was 4.5 points out of a possible 5 points. (Fig. 2.6).

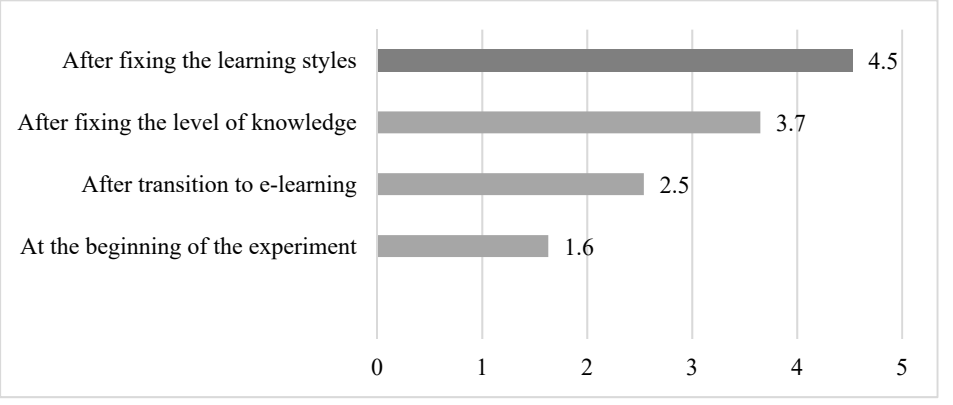

Figure 2.6 Computer Science Basics topics comprehension after the third step of the experiment

To sum up it should be noted that the feedback, received from the students' groups, was also different. The *test* group shows higher motivation for further learning compared to the *reference* group.

Similar directions in personalization of learning and teaching in electronic educational environment of the university can be found in article [73]. The source discusses issues about the pedagogical design of electronic educational environment of the university. The criteria for the design of a pedagogical scenario for each learning and teaching style are defined and described there.

2.5 Conclusion and discussion

Present chapter presented to readers three stages of the experiment with the Computer Science Basics course modification: "*e-course*", "*prior knowledge*" and "*learning styles*". Named steps include such kind of innovations as:

- the course transference to Moodle electronic environment;
- training materials update and new materials creation;
- teaching students in accordance with their prior knowledge and learning preferences;
- new teaching strategies.

During the experiment, there was a positive dynamics in the students' understanding of the subject. This fact shows the correctness of the chosen methodology. Furthermore, this correctness was confirmed by results of the examination among the students of the two groups, test and reference.

As a result of the described experiment the author of the thesis and author's colleagues got the first part of the new Computer Science Basics course for first year non-IT students. Chosen strategy and techniques have allowed educators to teach students more effectively and learners got the opportunity to grasp new knowledge more quickly and systematically.

Classroom activities of teachers and students took place in mutual communication. Therefore, the guidance and the formative role of the teacher of the renewed Computer Science course was realized in the creation and review of the theoretical material and the material in practical classes.

During the process of upgrading the Computer Science Basics course, the author has created the model of individualization of the educational process in an e-environment, which considers the level of students' prior knowledge and their preferences in the learning process (Fig. 2.7). On the author opinion, this approach to the individualization of learning was applied for the first time.

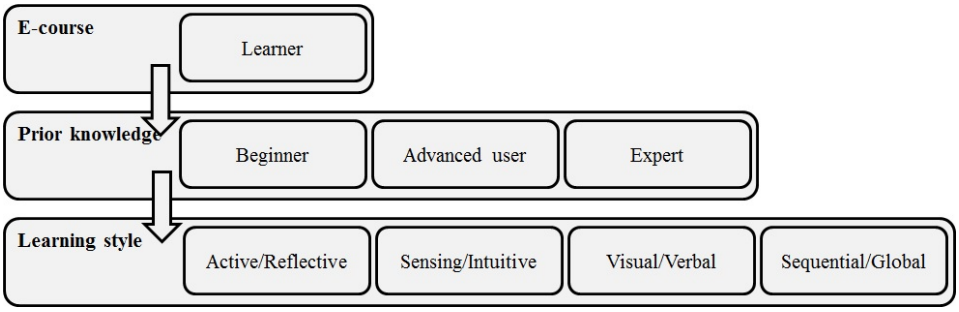

Figure 2.7 The model of the course individualization

Applying this model, the author of this thesis and the Computer Science Basics course instructors try to find an individual approach to each student in the e-course and make it more dynamic, flexible and attractive for first year non-IT students.

The next chapter provides readers with statistical analysis of the described experiment and the result, which once again confirms the correctness of the chosen direction in the current research.

3 STATISTICAL ANALYSIS

In the previous chapter innovation stages of the first part of the Computer Science Basics course were described. Throughout the experiment, positive students' feedback and good exam results showed the positive effect of the course and curriculum modifications. Finally, it was decided to examine the data with statistical methods with the aim to check and demonstrate the correctness of the chosen approach to an educational process. The main aim of current chapter is statistical analysis of the experiment results.

This chapter is based on the paper B (*Mironova, O.; Amitan, I.; Vendelin, J.; Saar, M.; Rüütmann, T. Strategies for the Individualization of an Informatics Course.*) and C (*Mironova, O.; Amitan, I.; Vendelin, J.; Vilipõld, J.; Saar, M. Maximizing and personalizing e-learning support for students with different backgrounds and preferences.*)

3.1 The Method of Analysis - Student's t-test

As the method of the hypothesis testing, the author chooses Student's t-test [74] for comparison of two means. This statistical method is used to see if two sets of data differ significantly. This test assumes a normal distribution of samples and not significant differences between the standard deviations of either samples.

The testing aim is to show that there were no significant differences between the *test* and *reference* group students in September, while in January the results are significantly different.

3.2 The Formulae Used

For calculations the author uses the formulae for the averages $\bar{x}$ (1) and corresponding standard deviation S for both groups (2):

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{1}$$

$$S = \sqrt{\frac{\sum_{i=1}^{n} \left(\bar{x} - x_i\right)^2}{n-1}} \tag{2}$$

After that, the standard error σ was calculated using the formula 3.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left(\bar{x}_{Test} - x_{iTest}\right)^2 + \sum_{i=1}^{n} \left(\bar{x}_{Ref} - x_{iRef}\right)^2}{n \cdot (n-1)}} \tag{3}$$

Finally, using formula 4 author calculates the experimental value $t_{\exp}$:

$$t_{\exp} = \frac{\left|\overline{x}_{Test} - \overline{x}_{Ref}\right|}{\sigma} \tag{4}$$

In addition, to compare this value with theory we need to calculate the degree of freedom df using formula 5:

$$df = 2n - 2 \tag{5}$$

3.3 The Initial Data

As initial data for calculations, the author chose September 2013 and 2014 students' test results of the *test* and *reference* groups, and the same groups' results in January 2014 and 2015. All numerical results are presented in the Appendix of the current thesis.

It should be mentioned here that students of those two group, *test* and *reference*, solved the same test with similar questions at the beginning and at the end of the semester.

All the considered samples are distributed normally (Fig. 3.1 – Fig. 3.4).

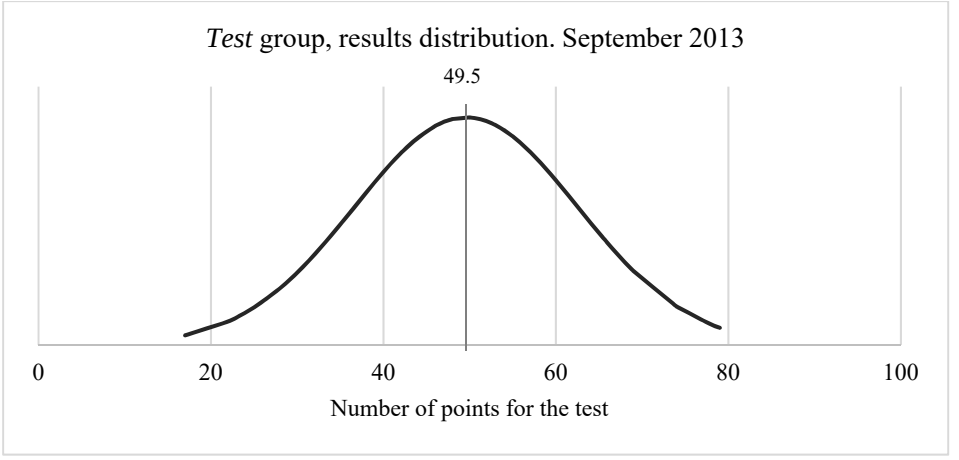

Figure 3.1 Test group students' test results in September 2013

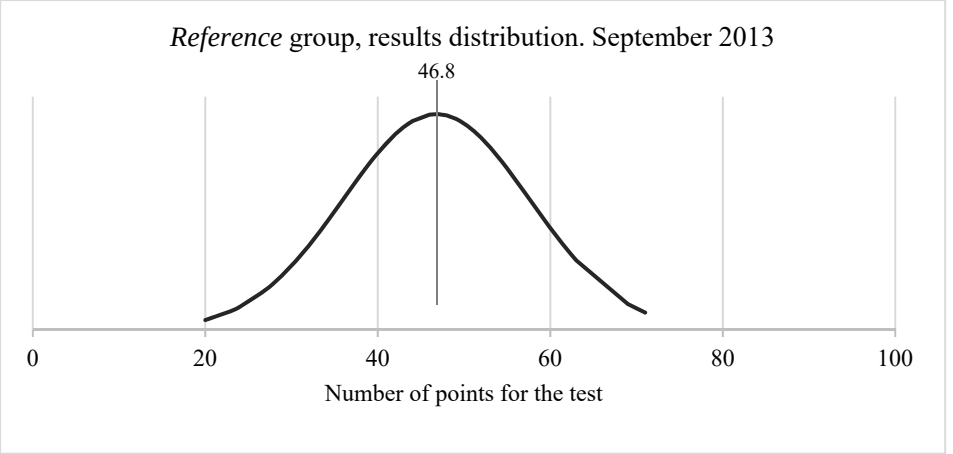

Figure 3.2 Reference group students' test results in September 2013

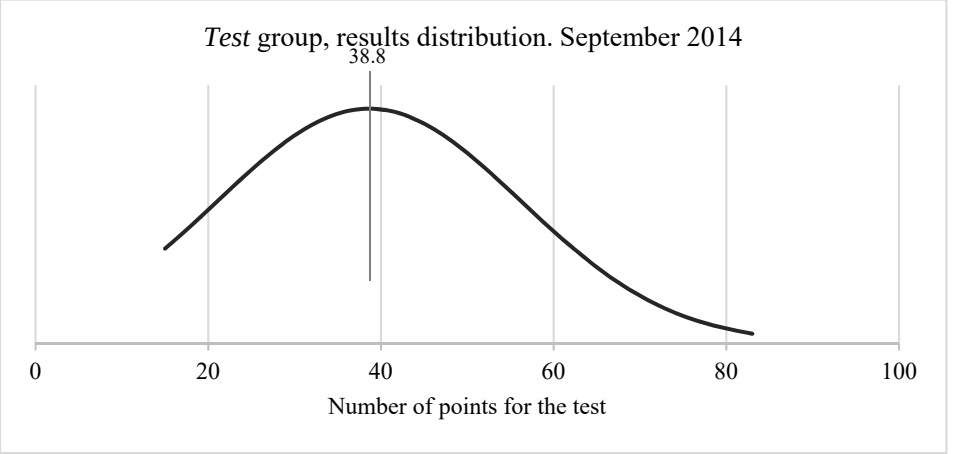

Figure 3.3 Test group students' test results in September 2014

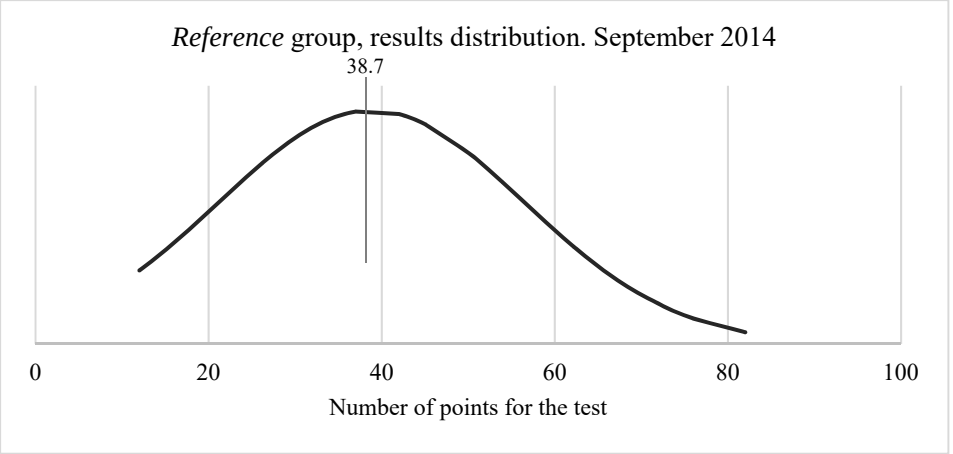

Figure 3.4 Reference group students' test results in September 2014

Table 3.1 Both groups' results and sizes

	September 2013		September 2014	
	Reference group	*Test* group	*Reference* group	*Test* group
Test results (points)	Number of students			
0 - 59	72	79	62	68
60 - 84	17	10	13	7
85 - 100	0	0	0	0
Samples sizes (n)	89	89	75	75

3.4 Calculations

The means $\bar{x}$ and the corresponding standard deviations s are calculated by using the formulae 1 and 2:

Table 3.2 The means and standard deviations in Septembers

	September 2013	September 2014
$\bar{x}_{Test}$	$\frac{4413}{89} \approx 49.6$	$\frac{2898}{75} \approx 38.7$
$\bar{x}_{Reference}$	$\frac{4164}{89} \approx 46.8$	$\frac{2912}{75} \approx 38.8$
S_{Test}	$\sqrt{\frac{10222.94}{89-1}} \approx 10.78$	$\sqrt{\frac{22793.28}{75-1}} \approx 17.55$
$S_{Reference}$	$\sqrt{\frac{14721.62}{89-1}} \approx 12.93$	$\sqrt{\frac{23010.75}{75-1}} \approx 17.63$

Now it can be seen see that there is no significant difference between the standard deviations in either groups. It means that it is possible to continue with Student tests.

The standard error of the difference between the two means is calculated by using formula 3:

Table 3.3 The standard errors of the difference between the two means

	September 2013	September 2014
σ	$\sqrt{\frac{10222.94+14721.62}{89\cdot(89-1)}} \approx 1.79$	$\sqrt{\frac{22793.28+23010.75}{75\cdot(75-1)}} \approx 2.87$

Experimental t value is calculated using formula 4:

Table 3.4 Experimental t values

	September 2013	September 2014
$t_{\exp}$	$\frac{\lvert 46.79-49.58\rvert}{1.79} \approx 1.56$	$\frac{\lvert 38.64-38.83\rvert}{2.87} \approx 0.07$

To compare this value with the theoretical t_{th} we need to calculate the degree of freedom using formula 5:

Table 3.5 The degrees of freedom

	September 2013	September 2014
df	$2 \cdot 89 - 2 = 176$	$df = 2 \cdot 75 - 2 = 148$

Using formulae 1 to 4 the author then calculated both groups' results in January 2014 and 2015:

Table 3.6 Both groups' results in Januaries

	January 2014	January 2015
$\bar{x}_{Test}$	$\frac{8029}{89} \approx 90.2$	$\frac{6687}{75} \approx 89.2$
$\bar{x}_{Reference}$	$\frac{6896}{89} \approx 77.4$	$\frac{5621}{75} \approx 75.0$
S_{Test}	$\sqrt{\frac{5092.94}{89-1}} \approx 7.61$	$\sqrt{\frac{6622.08}{75-1}} \approx 9.46$
$S_{Reference}$	$\sqrt{\frac{12518.22}{89-1}} \approx 11.93$	$\sqrt{\frac{12161.79}{75-1}} \approx 12.82$
σ	$\sqrt{\frac{5092.94+12518.22}{89\cdot(89-1)}} \approx 1.50$	$\sigma = \sqrt{\frac{662208+1216179}{75\cdot(75-1)}} \approx 1.84$
$t_{\exp}$	$\frlac{\lvert 90.21-77.44\rvert}{1.50} \approx 8.49$	$\frac{\lvert 89.16-74.95\rvert}{1.84} \approx 7.73$

3.5 Numerical Results of the Experiment

Using the table of theoretical t_{th} values with the corresponding degree of freedom (Table 3 in the Appendix), it was founded that the means of September's results are not different at any critical level (Table 3.7 The means in Septembers):

Table 3.7 The means in Septembers

September 2013	September 2014

$t_{exp} < t_{th}$ $1.56 < 1.65$	$t_{exp} < t_{th}$ $0.07 < 1.65$

This means that at the beginning of the course both groups of students, the test and reference, had the same level of knowledge.

January 2015 results are the opposite – the means are different at critical levels.

Table 3.8 The means in Januaries

January 2014	January 2015
$t_{exp\ Jan} > t_{th}$ $8.49 > 3.29$	$t_{exp\ Jan} > t_{th}$ $7.73 > 3.29$

These results show that the students who were taught using the presented system of the learning process individualization obtained knowledge much better than the others did.

The progress of both groups is graphically shown in the Figure 3.5 - Figure 3.8.

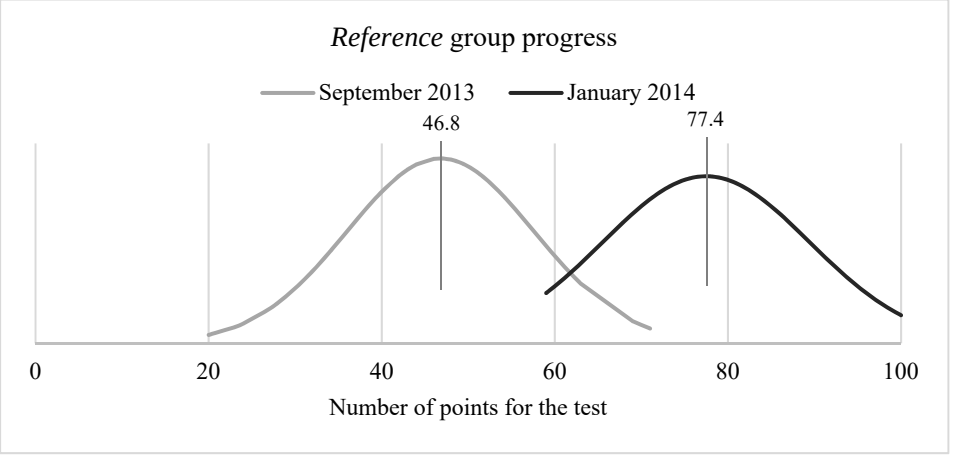

Figure 3.5 The reference group progress in 2013/2014

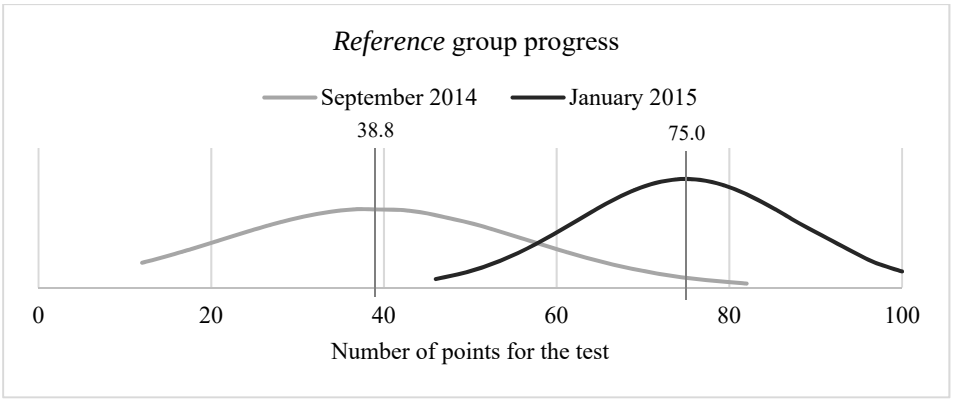

Figure 3.6 The reference group progress in 2014/2015

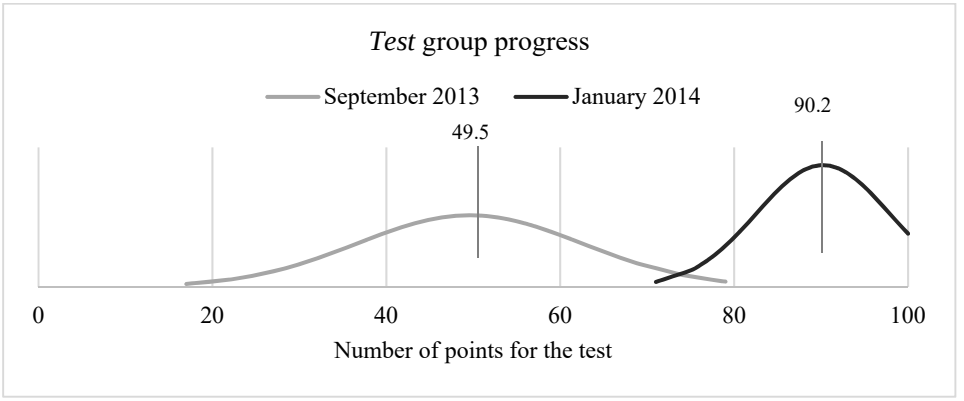

Figure 3.7 The test group progress in 2013/2014

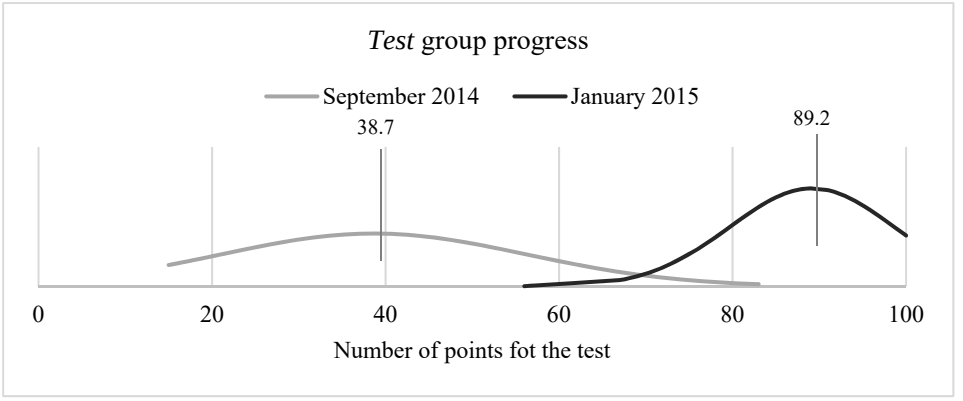

Figure 3.8 The test group progress in 2014/2015

In addition, the progress of both groups is demonstrated in the Table 3.9 and Table 3.10 and in the Figure 3.9 - 3.12:

Table 3.9 The reference groups progress

	Reference group			
	September 2013	January 2014	September 2014	January 2015
Test results (points)	Number of students			
0-59	72	1	62	9
60-84	17	68	13	47
85-100	0	20	0	19
Total	89	89	75	75

Table 3.10 The test groups progress

	Test group			
	September 2013	January 2014	September 2014	January 2015
Test results (points)	Number of students			
0-59	79	0	68	1
60-84	10	21	7	20
85-100	0	68	0	54
Total	89	89	75	75

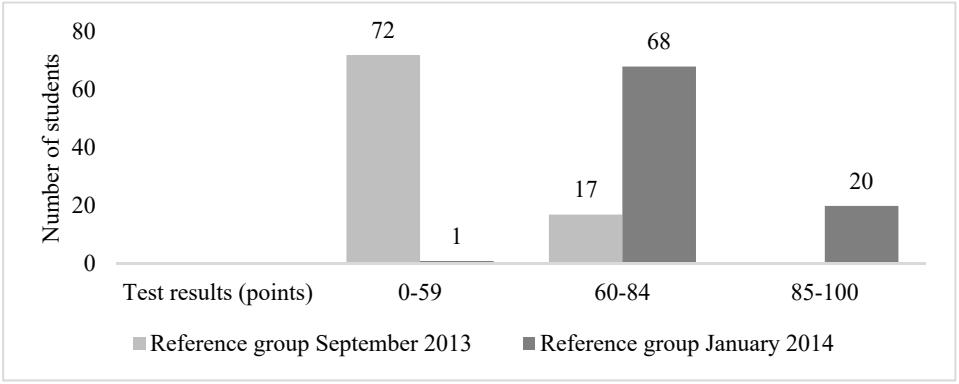

Figure 3.9 The reference group progress in 2013/2014

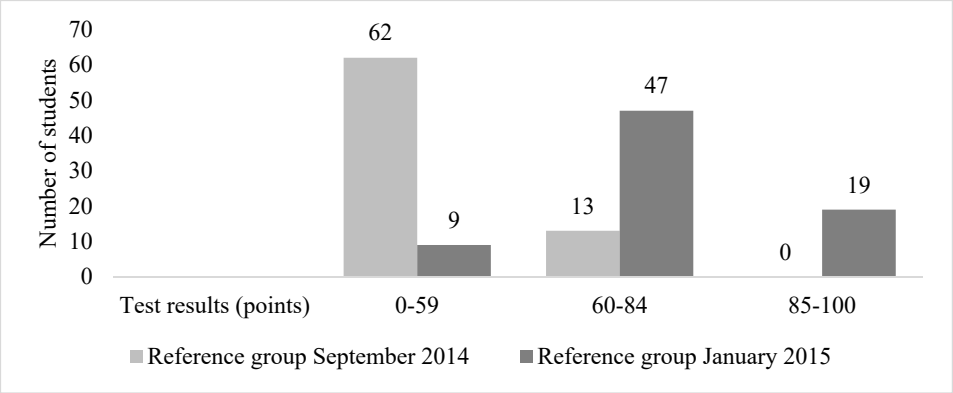

Figure 3.10 The reference group progress in 2014/2015

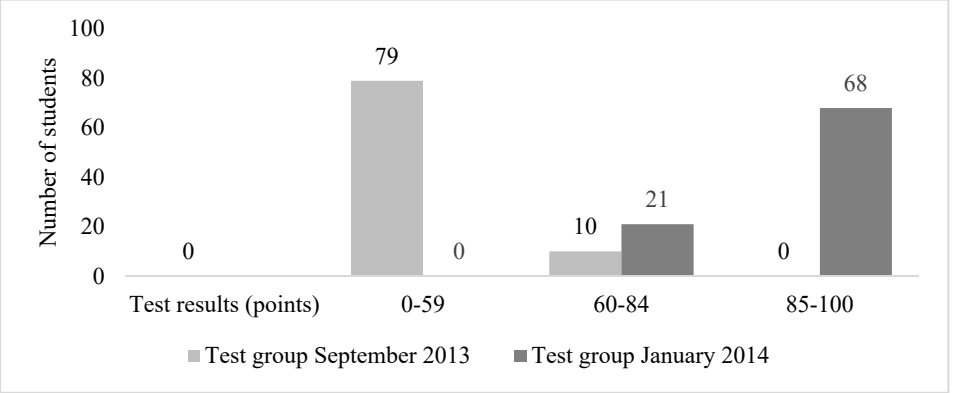

Figure 3.11 The test group progress in 2013/2014

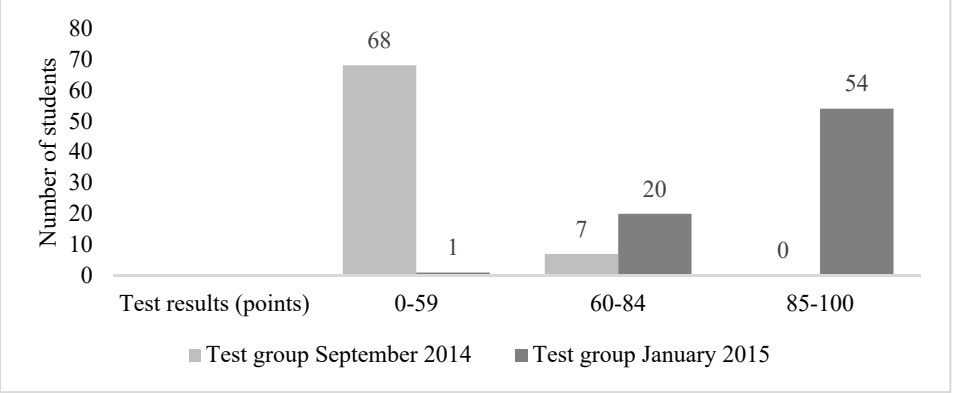

Figure 3.12 The test group progress in 2014/2015

These results confirm the validity of current study and the chosen method of course material and learning individualization.

3.6 Conclusion and discussion

The statistical analysis presented in the current part of the thesis and its numerical results show positive outcomes of the strategy used.

The calculations with two relevant groups, the *test* and *reference*, demonstrate significant differences in achievements at the end of the first part of the Computer Science course.

In the next chapter the author attempts to solve main issues related with difficulties, which non-IT beginners face in the second part of the Computer Science Basics course. This course section focuses on the programming basics and usually is sophisticated for the majority of the non-IT first year students.

4 PROGRAMMING FOR BEGINNERS

As was mentioned previously, in *Chapter 1*, the Computer Science Basics course for non-IT first year students consists of two parts. Accordingly, it is necessary to improve and develop the second part, programming basics, in order to make it more affordable and attractive for the audience. Considering this improving it is necessary one more time to note that this part of the course is particularly difficult for non-IT first year students.

The difficulties in teaching programming have been described and systematized in researches [75, 76].

Due to the fact that all first year non-IT students, who learn the Computer Science Basics course have a minimum level of knowledge in programming field, it was decided to change the teaching strategy and first of all concentrate on the algorithmization and modelling rather than coding.

As experience shows, the main programming concepts that are complicated for the non-IT learners' understanding are:

- object and its properties and methods;
- events;
- parallel and sequential processing;
- data: variables and lists;
- conditional statements;
- iterations (cycles);
- subroutines.

At the same time, they are the most important concepts in programming and there is no opportunity to learn and teach without them.

Present chapter is based on papers D (*Mironova, O.; Amitan, I.; Vendelin, J.; Vilipõld, J.; Saar, M. Object-Oriented Programming for non-IT Students: Starting from Scratch*) and E (*Mironova, O.; Amitan, I.; Vilipõld, J.; Saar, M. Active learning methods in programming for non-IT students*).

4.1 The Programming Introductory Tool

As was mentioned in article [77], the key challenge in learning programming is to acquire different sets of skills at the same time. In recent years, the demand for programmers and student interest in programming have grown rapidly, and introductory programming courses have become increasingly popular. Learning to program is hard however [78].

As practice and experience show, for better comprehension, it is good to graphically demonstrate and provide an opportunity to try out things that are hard to understand. This is especially important for *visual* and *active* learners with

strong preferences who are the main part of the audience on the Computer Science Basics course. Their numbers are demonstrated in the Table 4.1.

Table 4.1 Number of students with strong learning preferences

	2012/13	2013/14	2014/15	2015/16
Total	**156**	**178**	**150**	**168**
Visual learners	36	62	57	67
Active learners	27	16	16	21

A new and rapidly upcoming way in teaching programming basics is visual programming using an environment, which has been created especially for learning. The most popular are aforementioned Scratch, Snap! [79], Blockly [80] graphical tools, which are usable via browser. They make the programming process much easier for the beginners, especially for non-IT, who have not any experience in building algorithms, programming and coding.

Prominent advantages of using visual programming, like motivation, interest and enthusiasm during the study, this thesis reader can found in different sources [81 - 83].

Regardless of the students' age, Scratch greatly helps to grasp the basics and develop obtained knowledge. The research [84] results also showed that students could successfully learn important concepts of Computer Science, although there were problems with some concepts such as repeated execution, variables, and concurrency.

Moreover, Scratch gives advanced users the opportunity to extend the vocabulary through custom programming blocks written in JavaScript [85].

In the presented Computer Science Basics course the graphical programming environment Scratch is used as a supporting tool before VBA or Python. After a few years of practice, the course instructors came to the conclusion that Scratch is an effective introductory tool to understand both the object-oriented approach and the functionality of a program. This conclusion is reinforced by some facts:

- syntax errors are impossible in Scratch;
- Scratch works as an interpreter;
- graphical command blocks give an excellent visual picture of the different controls, used in the program;
- Scratch is simple and expressive, it makes understanding the behaviour of created objects easier.

The author of this doctoral thesis has founded the support in chosen approach to programming basics in [86]. The main idea of this approach is a *visualization* of any program. Without any doubt, visualization plays an important role in teaching programming basics for novices. The author of this thesis is totally agrees with this statement.

However, it should be noted that in present research Scratch visual programming environment is considered as a basic tool for visualization. Thus, Scratch performs two roles in described teaching approach: new programming language, which is used for introduction to textual programming and after, on the next stage, it is the instrument for the explanation and visualization.

In this chapter, the author discusses the design principles that guided her development of Scratch and her strategies for making programming accessible and engaging for everyone.

With regard to object-oriented programming, creation of objects in Scratch is provided by drawing or importing graphics. Scratch objects are named sprite and each one has its own properties and methods. Combining the blocks for each object creates the methods. Some of the blocks are used to show the reaction of the object to some events. Thus, it can be seen here the main aspects of object-oriented programming resulting in an attractive animation.

Brennan and Resnick [87] have identified seven concepts with examples, which people tend to use when they program in Scratch:

1. Sequences;
2. Loops;
3. Events;
4. Parallelism;
5. Conditionals;
6. Operators;
7. Data.

Based on these concepts and personal experience in programming teaching let the author demonstrate how Scratch solves issues in the problem areas in programming basics, which were mentioned above:

4.1.1 Object with its Properties and Methods

Each Scratch object named Sprite has properties like the name, coordinates, direction, rotation style, etc. (Fig. 4.1).

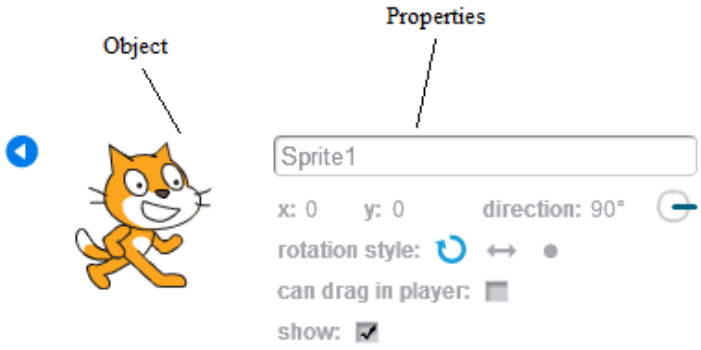

Figure 4.1 A Scratch object and its properties

Most of such properties can change their values by some method (script in Scratch) or a user may change them manually.

Figure 4.2 shows some blocks used to change properties of an object.

Figure 4.2 Blocks for changing the properties of an object

These blocks make it quite easy to understand the property concept.

4.1.2 Events

There is a rather long list of pre-defined events that can be handled by Scratch scripts. An event is handled by a script (method) starting with a special block. Some of the event blocks are shown in Figure 4.3.

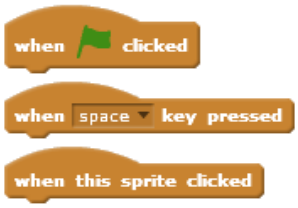

Figure 4.3 Some blocks to start the event handling script

Using an event block in the beginning of the script students had better understand how are organized the application and the relations between scripts.

4.1.3 Parallel and Sequential Processing

Running each script is considered a separate process. Scripts that handle the same event will be running in parallel after the event occurs. For example, the left scripts in Figure 4.4 are performing at the same time and they implement the reaction on the "click" event. The right script is the same actions but they are executed sequentially.

Figure 4.4 Realization of the parallel and sequential processes

It should be noted that it is easy to follow and explain the difference due to animation: the right figure side – the object firstly jumps and then moves; the left side – the object is jumping and moving at the same time.

4.1.4 Data

Variable is one of the basic concepts in all programming languages. Its meaning in programming differs from its use in mathematics, which first year students know from secondary school. As to lists, and especially indexation of the list elements, these are absolutely new concepts for the first year non-IT students and usually cause learning difficulties and mistakes in usage.

The graphical environment Scratch provides clarity in understanding the meaning of a variable and list concepts. All variables and lists have to be created manually before using them in a script. Using the command "Make a Variable" or "Make a List" in the Data group (Fig. 4.5) of the blocks students try out, see and this way understand it better as a named place in the computer memory.

The approach is very useful and worth considering in further programming activities. Thereby students become familiar with the programming

term “variable declaration”, which, in its turn, will be used later, during a programming in VBA.

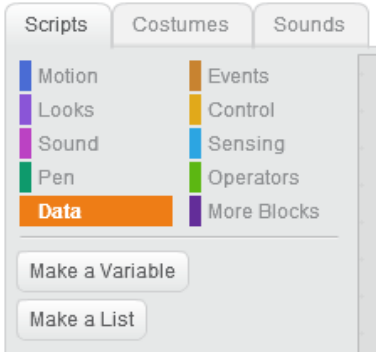

Figure 4.5 Commands of a Data group

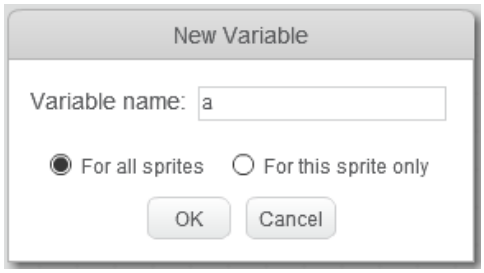

Figure 4.6 A variable declaration in Scratch

Furthermore, the users have to define the scope of the created variable or list, which leads them to understanding the meanings of global (in Scratch “For all sprites”) and local variables (in Scratch “For this sprite only”) and demonstrates the difference between these two terms (Fig. 4.6). If an object (sprite in Scratch) is active, only global and local variables and lists can be used - it can be seen immediately. Thereby students obtain the concept of the data scope faster and better.

A variable and list images on Scratch stage (Fig. 4.7) give students an overview of their values and some properties (e.g. length), which can be changed manually and/or in a program.

In addition, due to Scratch animation, learners can follow how program processes list elements – during the process indexes of the elements are blinking. Terms “list element” and “element index” become clear too because they are visible.

Moreover, there are two ways to fill a list with elements or clear a list: manually and in the program. This greatly helps students in using lists in other programing languages.

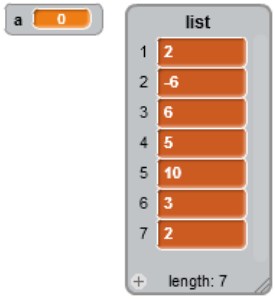

Figure 4.7 A variable and list images on Scratch stage

It should be also noted that learners have to define the scope of the created variable or list, which leads them to better understanding the meanings of global and local variables and demonstrates the difference between these two.

4.1.5 Conditional Statements

Using branching blocks in Scratch helps students to compose if-sentences. For example, it becomes clear why they have not written the second condition in them (Fig. 4.8).

Figure 4.8 The branching in Scratch

4.1.6 Iterations

Using Scratch blocks makes it easy to show students how iterations work – after its implementation learners immediately see the result: multiple iterations of chosen blocks (Fig. 4.9).

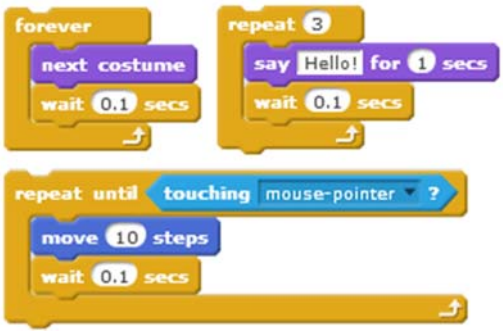

Figure 4.9 The iterations in Scratch

This construction of the iteration and branching blocks gives a clear picture about the actions inside these blocks.

4.1.7 Subroutines

The ability to split a big task into smaller pieces plays an important role in structured programming. This is the most preferred approach in building programs.

The majority of algorithmic languages support the definition of subroutines and functions, used in creating the code for the pieces of the project. Procedures without parameters provide an initial idea. One of the main methods of transferring data to subroutines is using the parameters. The authors' experience shows that this is the most confusing topic for a beginner.

Scratch 2.0, the new version of Scratch, provides teachers and learners with a perfect opportunity to make this issue easier. Learners can create and use their own Scratch blocks, where the definition of parameters is included (Fig. 4.10).

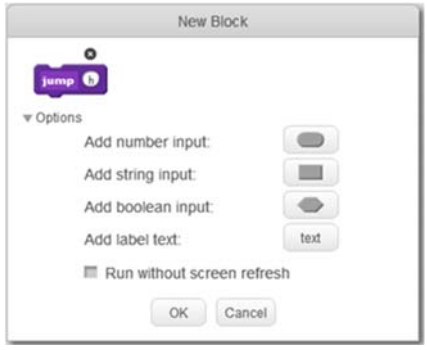

Figure 4.10 The user's block creation

Students learn to create a clear structure in their project. They divide their task into logical parts and create necessary user blocks, providing them with parameters. Now the main script can use standard and user-defined blocks, transferring the necessary data by means of parameters. Figure 4.11 shows the definition and calling of the user-defined blocks.

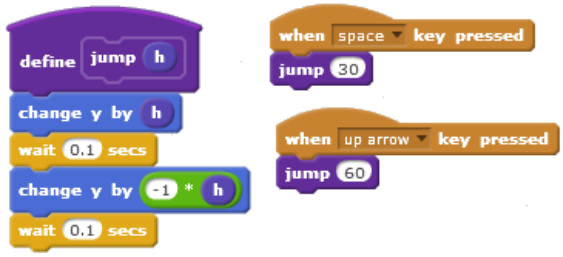

Figure 4.11 The user-defined block and its use in the main script

4.2 Coding Based on Scratch

The next Computer Science Basics course step is to proceed with other programming systems. Some lessons of practicing with Scratch tools make this transition easier.

It should be mentioned that according the annual students' feedback Scratch is the most popular module in the course. Students emphasize that just Scratch gives them an overview of the model of the problem and ways of its solution.

The similar approach to current "*coding based on Scratch*" can be founded in the doctoral thesis [88] that formulates and evaluates a pedagogical technique whose goal is to help beginners learn the basics of computer programming. However, this visual environment should not be considered only as an introduction to programming. It must be recognized as an excellent tool for presenting algorithms and models. Despite the fact that Scratch was not created to solve complicated tasks, it can surely play two roles in educational process: *introduction* to programming and *visualizing* the algorithms and models in textual coding. Its second role is especially important for non-IT learners and beginners.

It should be noted that in the current teaching approach, aimed to non-IT novices, the choice of the programming language is not as important as the development of students' algorithmic thinking and ability to build models. Based on Scratch learners get the holistic picture of the issue and can follow the algorithm. These skills are very useful for learners throughout their study and in future work.

4.2.1 Python

Python is an open source high-level computer programming language, developed in the end of the 80's by Guido Van Rossun in the Netherlands. Python is not one of the most widely used languages today, but statistics indicates that it is among the top ten languages [89].

A programming language for novices must enable the novice to learn these concepts without interference from the details of the programming language [90]. Python supports structured programming and procedural styles. This language does not require any declaration of simple variables, which makes work with it easier for the beginners. It has a large standard library. It is an open source and is available to all students. The language is a high-level language and has syntax, which allows programmers to express concepts in fewer lines of code than would be possible in some other languages.

Discussion about Python as an introductive programming language in high school can be found in [91]. Switch to Python for all first year Tartu University students and its success is described in [92]. Research [93] shows that due to Python having a simple syntax, if compared to other languages, it was easier to introduce programming concepts to High-School students, emphasizing programming and problem solution. Article [94] suggests that Python is an excellent choice for teaching an object-oriented Computer Science. Authors of the have been very pleased with the use of Python, finding that it affords a clear, coherent, and consistent presentation of object-oriented programming.

As practice shows, during programming in Python beginners usually face problems with indentations in their code. These indentations play a great role in Python and their misapplication can ruin the entire program. And again, Scratch helps to solve this problem: command blocks, which are situated inside iteration blocks and if-blocks, have the same indentation level as commands in Python (Fig. 4.12).

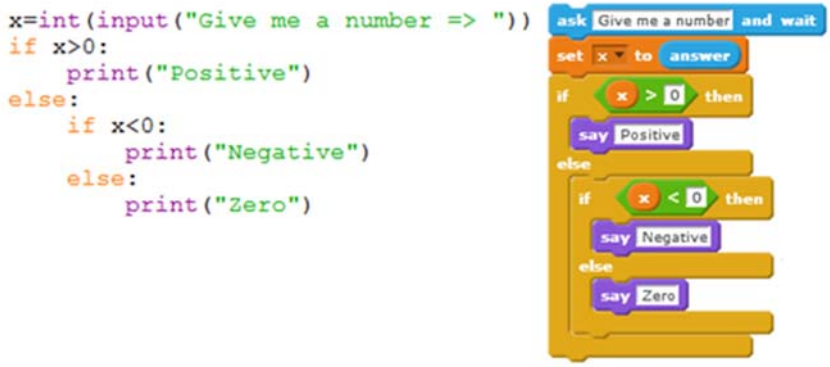

Figure 4.12 The indentations in Python and Scratch

The course main Python topics are covered in the textbook [95]. Similar ideas about Scratch and Python can be observed in [96].

4.2.2 Visual Basic for Applications

As was mentioned earlier, Scratch and Visual Basic for Applications (VBA) have a lot in common. For example, a number of graphic objects can be placed into MS Excel applications. There are a number of VBA methods, which have equivalents in the form of Scratch blocks (e.g. set colour). Therefore, the coding part of the course starts with graphical objects animation in Excel. After that students solve tasks related to their speciality.

During the second part of the Computer Science Basics course, the course teachers provide beginners with some ready-made procedures, to be used as "black-boxes". For example, procedures are used that correspond to Scratch blocks such as "wait" (Fig. 4.13), "move", "glide", "touching" etc.

```
Sub pause(pp)
  Dim pl
  pl = Timer() + pp
  Do
    DoEvents
  Loop While Timer() < pl
End Sub
```

Figure 4.13 VBA procedure and analogous Scratch block

This enables a faster transition to more sophisticated tasks in VBA. If students build and understand an algorithm (verbally, on paper and/or using Scratch), they can compose the textual solution using the existing procedures and their own commands. The course main VBA topics are covered in the textbook [97].

In addition, it should be noted that writing a program in VBA using already mentioned indentations is very useful and effective for beginners to understand better the program structure and better represent the result (Fig. 4.14).

```
x = InputBox("Give me a number")
If x > 0 Then
   MsgBox "Positive"
Else
    If x < 0 Then
      MsgBox "Negative"
    Else
      MsgBox "Zero"
    End If
End If
```

Figure 4.14 The indentations in VBA

4.3 Modelling and Algorithmization

Thus, the visual programming environment Scratch makes the first beginners' steps to programming field easier. Firstly, it motivates students due to its graphics and animation and, at the same time, helps them to follow the created objects´ behaviour. Secondly, it has not any syntax errors. Finally, Scratch is a remarkable tool for introducing a problem solution algorithm.

The Computer Science Basics course teachers always try to provide students' applications with a similar content. If a student has created a model in UML and the same application in Scratch, it is easier for him to "translate" it into the VBA or Python. Thus, if a learner understands the content of the model and the algorithm, it is easier also to understand the syntax of any language. The visual programming using flowchart is the best way for the beginners to create a program for general purpose [98].

The following example is about solving a typical textbook task where the program generates a random number and asks the user to guess it. Here we see the steps of solving the task: UML activity diagram (Fig. 4.15), the script from Scratch project implementing the behaviour of the cat (Fig. 4.16) and finally the program code in Python (Fig. 4.17) and VBA (Fig. 3.18).

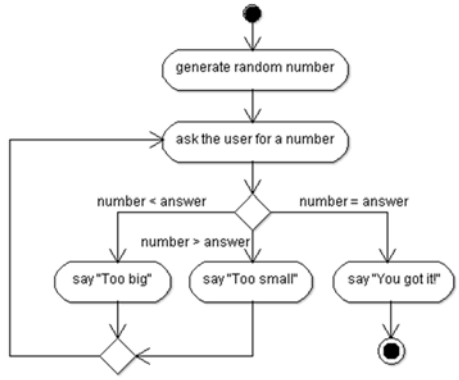

Figure 4.15 UML Activity diagram

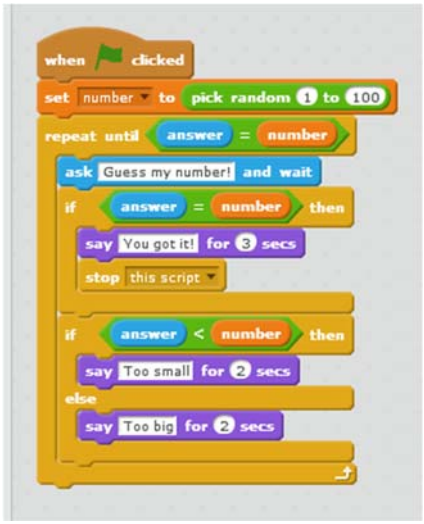

Figure 4.16 Scratch Project

```
from random import randint
number=randint(1,100)
while True:
     answer=int(input('Guess my number! '))
     if number==answer:
          break
     elif number>answer:
          print('Too small')
     else:
          print('Too big')
print('You got it!')
```

Figure 4.17 Python code

```
Sub GuessTheNumber()
Dim answer As Integer
Randomize
Number = Int(100 * Rnd + 1)
Do
    answer = InputBox("Guess my number!")
    If Number > answer Then
        MsgBox "Too small"
    ElseIf Number < answer Then
        MsgBox "Too big"
    End If
Loop Until Number = answer
MsgBox "You got it!"
End Sub
```

Figure 4.18 VBA code

4.4 Applied Active Teaching and Learning Methods

Various tests have been carried out to identify students' levels of prior knowledge in the second part of the Computer Science Basics course during 2012-2015. Unfortunately, all of them showed that non-IT students' level is *zero level*. On this basis, it was decided not to divide all the students according to their level of knowledge during the second semester but to offer them more visual materials and practical tasks based on their learning preferences.

Thereby one of the main objectives of teaching of programming basics to novices was to explain to students what is a model and an algorithm, and how they can be represented. In addition, it is necessary to mention the fact that most of the students were against the programming and claimed that this kind of knowledge is needless for them. Thus, teachers received one more task – to rework the programming module of the Computer Science course with the aim to increase students' motivation.

During the current research the author frequently mentions the term *motivation*. This is caused by the fact that the work on the improvement of the Computer Science course is aimed at students and their interest, which is inseparable from motivation. This is an abstract concept that is difficult to measure [99]. However, some general categories of motivation, such as intrinsic and extrinsic factors, were identified and measured [100].

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 [101, 102]. A research into the motivations of students for studying programming is described in the article [103]. Results raise a number of questions for the teaching of programming. Achievements of the study in learning motivation indicate that teachers need to provide opportunities for students to develop their thinking [104].

The best way to interest and involve students in the learning process – has been proved to be their active participation in it. It is necessary to mention here that active learning can support learners' development of the four capacities in many ways. For example, they can develop as:

- *successful learners* through using their imagination and creativity, tackling new experiences and learning from them, and developing important skills including literacy and numeracy through exploring and investigating while following their own interests;
- *confident individuals* through succeeding in their activities, having the satisfaction of a task accomplished, learning about bouncing back from setbacks, and dealing safely with risk;

- *responsible citizens* through encountering different ways of seeing the world, learning to share and give and take, learning to respect themselves and others, and taking part in making decisions;
- *effective contributors* through interacting together in leading or supporting roles, tackling problems, extending communication skills, taking part in sustained talking and thinking, and respecting the opinions of others [105].

In the current section of the thesis, relying on literature and personal experience, the author try to suggests some ways for making programming more engaging for non-IT learners with the aim of raising their interest.

4.4.1 The Visualization of the Processes

As mentioned above, the course teachers build UML activity diagrams to describe algorithms in complicated tasks. A verbal description and pseudo code are been using as well. Now, when Scratch projects are created as an introduction to programming, they can also be used to visualize, formulate and describe the problem.

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 program code according to the diagrams. Later on, students have to create the models themselves.

It has to be once more mentioned that according to tests most of the Computer Science Basics course students are visual learners. For them it is very important to see "how it works". Scratch, with its elements of attractiveness, helps those students to understand the main idea of creating an application.

VBA already has a built-in visualising tool: students can follow the code execution using the Locals Window (Fig. 4.19).

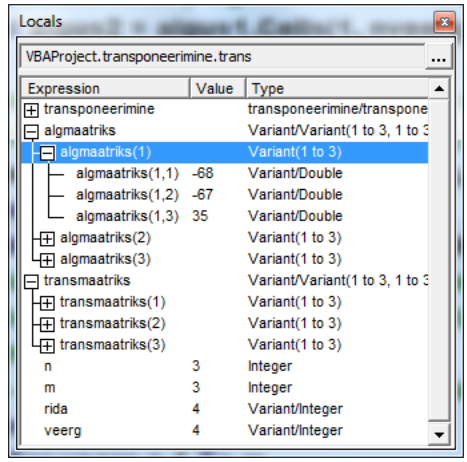

Figure 4.19 The Local Window in VBA

It automatically displays all the names of the declared variables in the current procedure, their types and their values. When the Locals Window is visible, it is automatically updated every time – students can see and check each step and its result in their program.

Python does not have such an opportunity, but still needs to be visualised. Computer Science Basics course teachers show students the Online Python Tutor [106]. Using the visualizing tool, the students can follow each step of their code and check the values and types of the variables, as well as the order of the operators during the execution. It has to be noted that this tool has some drawbacks, as it does not support the Python graphics, time functions and work with files. However, for the beginners in programming it gives the understanding of the code execution (Fig. 4.20)

Figure 4.20 Python online visualizing tool

4.4.2 E-learning

As was mentioned above, during the semester all students work in 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 Computer Science Basics course development and evolution the author of the thesis and lecturers who were involved in the experiment have been trying to adapt theoretical teaching materials for the e-environment and deliver it to students in most suitable forms (*Chapter 2*). The main 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.

After each new topic students have to fulfil corresponding tests. Modern testing systems provide the course instructors with a variety of test types and, using them the course the author 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 arranging 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' programs. This activity develops such an important skill as the ability to read and understand a code written by another person. Moodle environment provides teachers with this opportunity and they 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.

4.4.3 Face-to-face Lessons

Practice is one of the most important steps in learning the art of computer programming [107]. 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, the author has 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 Computer Science Basics course, the majority of the created applications are small games that students can play – one more motivating factor. Using Scratch,

students make games and within the process, learn to understand their algorithms. 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. Paper [108] suggests the effectiveness of similar approach to reduce initial difficulties for freshmen learning programming concepts.

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, proves, and demonstrates why one solution can be better and more logical than another can. However, there it is important to know what kind of learners are in the class. Therefore, it is possible to implement this kind of work only when the instructor is familiar with the audience.

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 are successfully applied, especially before practical tests, is correction of mistakes in a program text. As usual, students have their task description and corresponding 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 the author's experience shows, such assignments are useful if offered as pair work. Here the group work skills are developing among students and they learn to understand the program 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 more 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 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.

There should be noted that attendance affects the quality of knowledge and there are many reasons why it can facilitate grades, such as:

- lectures, class discussions, and other activities during class provide additional information beyond the textbook, class notes, and handouts;
- students can relearn information in class that they read in the textbook and handouts;
- students have increased opportunities to learn instructor's expectations and ask for clarification on assignments;
- students have increased opportunities to establish collaborative learning with classmates [109].

When new theoretical material is explained, the course teachers often use ready-made programs 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 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 the author applies in the Computer Science Basics 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.

4.5 Positive Outcome of Applied Teaching Techniques

As was mentioned above, programming basics for the first year students, who are non-interested in this field, is complicated. However, during the experiment with visual programming and active teaching, the course teachers have noticed the positive trend in learners' academic results. Average exam grades for last academic years are demonstrated in the Table 4.2 and in the Figure 4.21:

Table 4.2 Average exam grades during the experiment

Academic year	2010/11	2011/12	2012/13	2013/14	2014/15	2015/16
Average grade	2.5	2.7	3.0	4.0	4.2	4.3

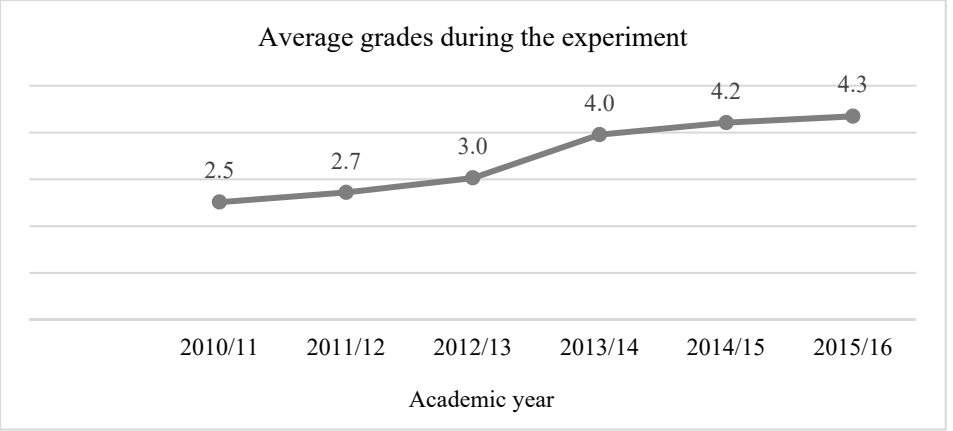

Figure 4.21 Average exam grades during the experiment

More detailed results, which were collected during the experiment, can be found in the Appendix of the current thesis.

Thus, it can be noticed that the visual environment Scratch and applied active teaching techniques really help beginners in their starting in a programming field. Scratch helps to grasp main concepts and keeps students' interest and motivation on the high level. In combination with the active teaching methods used, the course teachers managed to achieve good results in teaching programming basics.

4.6 'Basics of Applications Development and Programming' for School Pupils and Schoolteachers

Computer Science is necessary at modern schools for nowadays children. Results of the research [110] provide strong evidence to justify learning Computer Science in middle schools. The improvements in learning, teaching efficiency, and affective factors are easily discerned in the transition to secondary school Computer Science.

Speaking about programming for beginners, it should be noted that the non-mandatory course 'Basics of Applications Development and Programming' has been included in the curriculum of Estonian secondary schools starting from year 2011. The course was created based on the experience of Department of Software Science at TUT in teaching students for many years their main challenges in the beginning of the programming study. In considered course development authors and educators try to keep up with the times and trends in computer education as [111, 112].

This practical school course duration is 35 hours, basic tools there are Scratch, VBA and Python. The course consists of three components:

- methods and tools for creating applications;
- modelling and algorithmic principles;
- programming bases.

Unfortunately, there are no enough teachers for teaching this course in Estonia. It is the biggest problem in named course promotion.

In order to use and develop this new course and after get good results in teaching pupils the first need is to teach schoolteachers. At the present moment the majority of Estonian schoolteachers do not use knowledge of programming or very often do not have it.

Furthermore, primary and secondary school teachers are also very interested in using Scratch for teaching pupils but they have lack of knowledge and experience.

People often do a mistake that Scratch is only for programmers, teachers of Computer Science or for interested children. It is one more reason to show to schoolteachers how they can use Scratch for all school subjects for making their teaching process more attractive and dynamic for modern children [113].

That is why now TUT Department of Software Science is teaching Scratch for schoolteachers. This course is much deeper than Scratch courses for students because it has more methodical and didactic problems and tasks related to different subjects. Educators are using different methods of teaching, like group works, lessons-presentations and making curricula for future work.

Thus, the Department of Software Science at Tallinn University of Technology and the author of this thesis have created and constantly develop the following programming courses for beginners:

- 'Basics of Applications Development and Programming' for school pupils from 15 to 18 years;
- 'Basics of Applications Development and Programming' for schoolteachers;
- 'Programming basics' for children from 10 to 14 years.

The first of them is aimed to learners, who are already interested in programming and want to enhance their knowledge. The second one, for school teachers, is dedicated to didactic problems and their solutions. The main tools here are Scratch, VBA and Python. The volume of each of these components varies within 35 hours depending on the level of knowledge and skills, as well as the needs of learners. Scratch course for teachers and its outcomes are presented in [114]. Similar approach can be found in two-day workshop for teachers, which is described in article [115].

The author of this thesis suggests to teachers, who are new in the teaching of Computing, to read the article [116]. This study is able to offer guidance to teachers on how develop their Computing teaching skills. In addition, schoolteachers can familiarize with paper aimed to propose e-Learning model as an educational concept to teach introduction to programming for secondary school learners [117]. The objective of this model is to determine the components involved by applying collaborative learning in e-Learning, to enhance students' motivation and understanding in the subject Information and Communication Technology, specifically in topic introduction to programming. In addition, the author could suggest to Scratch teachers to join *ScratchEd* community [118]. This is an online community where Scratch educators share stories, exchange resources, ask questions and find people.

The last Programming Basics course is addressed to the younger generation with the aim to kindle their interest to IT and particularly to programming. Due to the age and level of knowledge learners of this course create applications only in visual environment Scratch.

For named programming courses new learning materials [119 - 121] were specially developed at the Department of Software Science at Tallinn University of Technology. The short report about Scratch in TUT this thesis reader can find in the presentation [122] from Scratch Conference 2013 [123].

For those learners who are interested in Scratch and want to develop their computational thinking, free/open-source web application *Dr. Scratch* [124] was designed in Spain [125]. Learners and instructors can use this tool to analyse Scratch projects and receive feedback on the quality of the programs [126].

4.7 Teaching Scratch

After TUT programming courses description it is necessary to give an overview of the several programming courses, which use Scratch for teaching. This overview should help readers to recognize the differences between them and the course, which is described in this thesis.

4.7.1 Scratch Courses in the World

Currently a variety of the programming courses for beginners is developed in the world. Some of them are free, some are not; some courses are online, some are using blended learning principles. They all are different, but programming using Scratch is their main principle. It is not possible to describe and analyse all the courses in the current research, however consider some of them.

Firstly, Harvard University new programming course CS50, which was launched in autumn 2015 [127]. This course considers C programming language foundations based on some Scratch blocks (Fig. 4-21 and Fig. 4-22), which help

to understand main programming principles. Scratch is used there during the first 2 weeks.

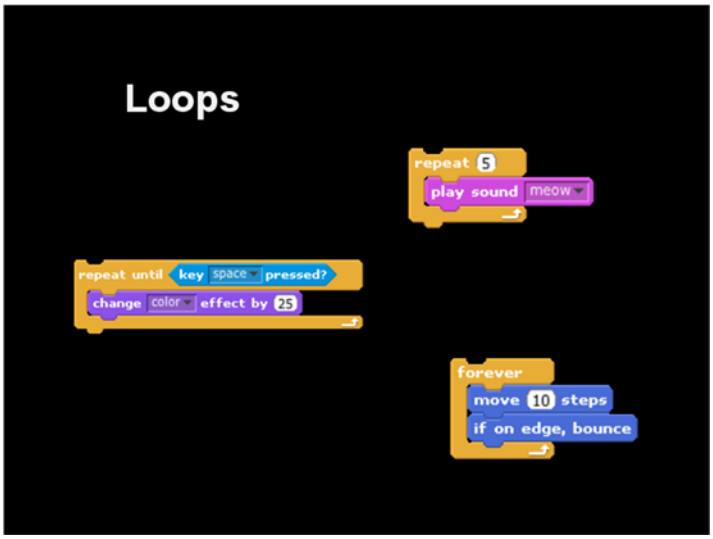

Figure 4.22 Loops with Scratch in CS50

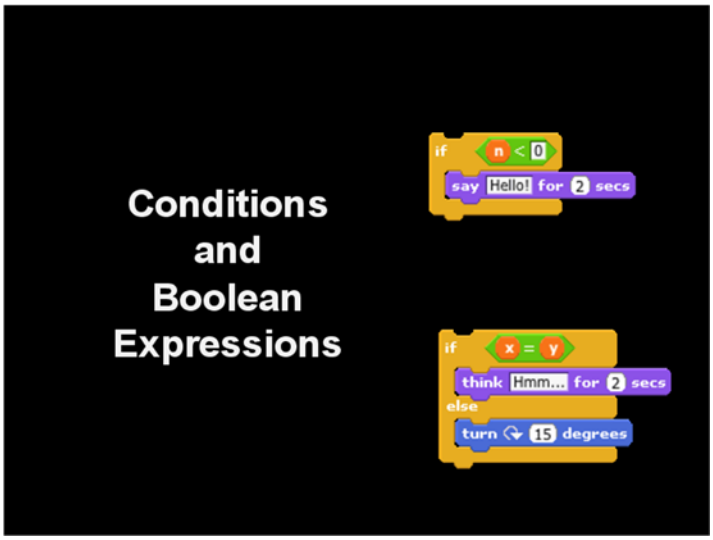

Figure 4.23 Conditions and Boolean Expressions with Scratch in CS50

Secondly ‘A Computer Science Principles Course’ [128], developed by the UTeach Institute at The University of Texas at Austin. Summary table [129] shows that one of the programming environments of this course is Scratch.

In addition is worth paying attention that Snap! - Scratch language implementation is one of the programming environments in the course 'The Beauty and Joy of Computing' developed at the University of California, Berkeley [130].

Free online short course – 'Scratch - Teach Computer Programming in Schools' [131] introduces learners to Scratch and its advantages in an educational setting, reviews the features and functions of the Scratch interface, and how to program using the Scratch software application. Programming course consists of 5 modules:

- Module 1: Introduction to Scratch;
- Module 2: Scratch Interface;
- Module 3: Scratch Tutorials;
- Module 4: Scratch Cards;
- Module 5: Scratch Assessment.

'Introduction to Programming with Scratch in Education' course [132] at the University of Northern Iowa takes learners through the ins and outs of programming with Scratch and prepare for introducing students to Scratch. The course schedule is:

Unit 1 - Getting Started with Scratch;

Unit 2 - Scratch for Storytelling;

Unit 3 - Scratch with Text Interaction;

Unit 4 - Scratch with mouse/keyboard interaction;

Unit 5 - Using Lists in Scratch.

The course 'Code Yourself! An Introduction to Programming' teaches learners how to program in Scratch, an easy to use visual programming language. More importantly, it introduces learners to the fundamental principles of computing and it helps them think like a software engineer [133].

One more course, which could be mentioned here, is free online course from Harvey Mudd College – 'Programming in Scratch' [134]. It should be noted that separate Python course [135] could be found there as well.

More various programming courses this thesis reader can find in: [136 - 139].

4.7.2 Scratch Courses in Estonia

In Estonia there are not so many programming courses for novices, which use Scratch.

The University of Tartu has developed Scratch teaching material for students [140]. This educational material consists of fourteen units, which cover the main Scratch topics. Short videos with explanations and practical assignments are used there.

In addition, there is one more 'Scratch' course [141], which consists of four parts with short videos, text materials and practical assignments.

Short report, which includes information about programming courses in Estonia can be founded in [142].

4.7.3 Analysis

The above-mentioned review shows that programming courses can be divided into two types:

- courses, which include Scratch as the *main programming tool*;
- courses, which include Scratch (more precisely, its elements) as the *introductive tool* to serious textual coding.

The first type has been mainly developed to meet the interests of younger students. No previous programming skills are required there. In these courses, elementary school students are introduced to fundamental programming concepts. Students learn how to create animations, computer games, and interactive projects using Scratch. Throughout these courses students create their own computer games and share them with their instructors and classmates.

The number of courses of the second type is far less. Those courses are aimed to university students and include some Scratch blocks for visual explanation of the basics programming concepts.

The teaching approach to the programming course, which is described in current thesis, is slightly different. Despite the fact that the course 'Basics of Applications Development and Programming' is also an introduction to programming the main aim there is to develop students' algorithmic thinking using certain programming tools.

The TUT programming course combines four programming environments: UML, Scratch, Python and VBA, which proportions vary depending on the age of learners, their level of knowledge and skills. Moreover, this course is able to be of interest to students of all ages from 10 years.

Practical assignments also vary depending on the audience, however the teaching algorithm is the same: students construct models of the tasks and try to solve them using aforementioned environments. This approach focuses mostly on the model, algorithm and their visualization, rather than teaching syntax and coding techniques. The basic principle here: *if a student is able to build a holistic*

model and create an algorithm, then student is able to implement it on any certain language.

4.8 Conclusion and discussion

In conclusion, it is necessary to note that success in programming basics learning and teaching to novices depends on many above considered aspects. Most important of them is the proper choice of starting point – the first programming language or programming environment. This language should raise pupils' interest, be useful in future learning and be a tool for the visualization. As practise shows, Scratch is a perfect tool for grasping modern concepts in building applications and it greatly helps students in their future study and professional work.

In addition, the use of a proper pedagogical technique is important at the stage of the beginning of the teaching programming. The correct method of teaching that includes active learning methods in combination with blended learning provides success in whole learning and teaching process.

Based on the above said, it should be concluded that in this thesis described methodology is based on mostly on the model, algorithm and their visualization, rather than teaching syntax and coding techniques. In this case, the choice of the programming language for a textual coding is up to the teacher, which is aware of the needs of students and monitors trends in programming languages.

The main aim of chosen approach is to develop students' algorithmic thinking and teach them how they can build a model and follow an algorithm of any problem. To achieve this goal the author and author's colleagues try to provide students' applications with a similar content using different programming systems.

Moreover, it is necessary to explain to the first-year students that knowledge and skills that they gain in this course are useful for them throughout the study and in future work. To achieve this goal the author and author's colleagues provide students' tasks and assignments that are close to real life.

Visualized tools like Scratch are a good way to introduce and, afterwards, better and faster understand the main concepts of object-oriented approach. In addition, Scratch makes it easier to write a programming code in any language when these concepts are already understood. After "the first level understanding", any programming process for non-IT students and especially for the beginners should be visualized in any case using suitable instruments.

CONCLUSIONS

This chapter answer the research questions, summarizes the main results of this thesis, and proposes the future work.

Answers to the Research Questions

During the study the author of this thesis could get the answers to the questions raised at the beginning of the experiment.

General research question was "*How to help novices to overcome complexity of Computer Science Basics learning?"* This question was divided into two sub-questions:

"How to combine study about learners' level of readiness with theory of learning styles?"

"How to use this combination in practice with the aim to individualize teaching and learning?"

Answers were identified during the study – new teaching approach was developed and successfully applied in practice.

- Firstly, students should have the opportunity to learn not only in classes.
    - Taking this fact into account course teaching and learning were transferred into Moodle e-environment that has given teachers and students more opportunities for effective teaching and successful learning.
- Secondly, non-IT students usually have not enough prior IT-knowledge before any Computer Science course. Moreover, TUT matriculants' level of computer skills has been steadily decreasing.
    - Solving this issue the author and author's colleagues, who were involved in the experiment, have found a solution – students division into groups according to their prior knowledge. This division has given course instructors an overview of students' level and accordingly their needs in additional learning materials. Applying this idea the author could provide all students with learning materials, which are suitable just for them.
- In third place, there is the known fact that all people learn in different ways. However, regardless of this all kind of learners should be provided with the best conditions to acquire knowledge.
    - Solving this problem the author has divided students into groups according to their learning preferences according to Richard Felder's theory of learning styles. This division has

provided students with appropriate learning materials and assignments.

- Fourthly, teaching programming foundations to students, who do not have prior knowledge and especially interest and motivation in named field, is complicated for teachers. At the same time such learning process is difficult for these students.
    - Solving the situation the author and author's colleagues have worked out and have applied the teaching system, which has greatly helped them and students in programming basics teaching and learning. The main idea of this approach is the *implementation of the visual programming before any textual coding*. In addition, the experiment has shown that the majority of modern students are visual and active learners – these types of learners need something dynamic and attractive. Students, who have been taught in this manner, understand programming basics better and more quickly than others do. Using Scratch they can create a model and an algorithm and after to translate it to VBA or/and Python. It is necessary to add here that VBA and Python have their own visualizing tools, which have been actively used in the second module of the Computer Science Basics course.
    - Moreover, Scratch has added some attractiveness to the programming course that has affected to students' interest and motivation.
    - In addition, during the practical part of this research, contact lessons, the author had the opportunity to implement active learning methods in practice and finally give recommendations on the use of appropriate forms, methods and organization of training.

The innovated teaching approach was able to increase students' interest and motivation, which is a very important aspect especially for the first year students at university. Students could overcome main above named difficulties and received better results.

Moreover, during the experiment variety of new teaching materials for different students' groups was developed. These materials can be used not only for university students. School pupils and schoolteachers, who are interested in Computer Science Basics and who are beginners in this field, can be taught using this educational set.

In addition, the author would like to emphasize again that the content of the material, which is adapted to students' prior knowledge level and to their

learning styles, helps them better deal with a learning material and is an important factor in learning individualizing.

Contributions

During the research, having received answers on the research questions, the author of this doctoral thesis has created blended teaching and learning model, which is based on students' prior knowledge and preferences in learning process and is realized in Moodle e-environment.

Through the teaching model, which was developed on the TUT Computer Science Basics course it was possible to deeply explore students' personal character traits and preferences using Richard M. Felder's learning styles theory and combine this significant information with important data about students' prior knowledge level. The operability and effectiveness of this combination were statistically confirmed during some years of the experiment.

Moreover, according to the experiment results, the author could individualize the teaching and learning processes and significantly improve their quality. Thus, it was possible to show the effectiveness of the theory of combining aforementioned teaching strategies.

Due to this combination blended teaching and learning model for Computer Science Basics course for beginners has been worked out. This course model provides the essential support for students with appropriate learning materials and activities and accordingly helps them to achieve higher level of knowledge.

The main attainment of developed model is effectiveness of teaching and accordingly success in learning, which were achieved by maximal teaching and learning flexibility and individualization. Such aspect has not been taken in the account to the necessary extent in previously considered researches. Teaching model, presented in this doctoral work, proceeds from each student's needs, preferences and learning style. Each student here has been considered as a set of certain parameters, which compile his/her individuality and determine the maximally effective route through training materials in the learning process.

To conclude it should be noted that the main novelty of this thesis is firstly combined two teaching strategies: teaching according with learners' prior knowledge level and learners' learning styles. Chosen methodology can be understood as an in-depth examination of the learners' data with the aim of extracting useful information from it to cater for flexible and more effective teaching and learning.

In reference to teaching programming to novices, this thesis author has presented the teaching technique, which is based on visual programming using Scratch. The novelty of presented approach is using of Scratch in teaching and

learning – it plays two significant roles here: the first and introductive programming language and the basic visualizing instrument for representing an algorithm during a textual coding.

The main idea of this strategy is to teach beginners in programming how to build a model of solving problem and create an algorithm of its solution using visual environment. When algorithm building using Scratch is clear for a learner and main programming principles were studied, it is the right moment for switching to textual coding.

Using Scratch students have the opportunity not only create a model, but also they learn such main programming concepts as conditional statements, iterations and data. Subsequently it is much easier to learn and apply a syntax of any particular programming language, having basic knowledge in the visual environment.

Thus, in the present research all shortcomings, which were discovered in the sources considered by the author [20 - 31], were taken into account:

- If students do not have any experience and motivation, it is necessary to start with something simple and attractive. Scratch here is the optimal solution.
- The optimal set of learning topics for the first year students was selected. This set fully satisfies the needs of students at the university and also gives them the necessary basis for the expansion of their knowledge.
- Difficulties in learning arise for different reasons. In this study, at least two reasons were eliminated: the lack of prior knowledge and difficulties with the perception of the learning material.
- Personalization of training was achieved by considering of the characteristics of each student.
- One of the distinguishing features of this study was the fact that students were not aware of the experiment. This condition also reinforced the reliability of it results.

Moreover, the presented approach to Computer Science Basics teaching to novices follows the main principles of computational thinking [2 – 4, 36] - students have been taught:

- to formulate problems in a way that enables to use a computer and other tools to help solve them;
- to organize data logically and analyse it;
- to represent data through models;
- to automate solutions through a series of ordered steps;
- to identify, analyse and implement possible solutions with the goal to achieve the most efficient and effective combination of steps;

- to generalize and transfer a problem solving process to a wide variety of problems.

Future Work

At the present time Computer Science contents develops and changes fast. New generation of learners needs contemporary and extensive knowledge in this field. In connection with this situation, the main goal of nowadays educators is to provide contemporary learners with such kind of knowledge.

Pedagogical sciences also do not stand still. Combining leading trends in both directions is possible to reach success in teaching providing students with high quality educational materials and learning activities.

Now the author of this thesis is head of the working group, which develops wider course, which name is IT Foundations and which includes variety of educational topics. The author intends to continue developing the created teaching approach for novices in the chosen style and direction, trying to adapt it maximally to learners with different preferences as much as possible.

In addition, using the worked out programming teaching strategy, the author continues to develop and promote programming basics courses for beginners and all learners who are interested in the acquisition of new knowledge. The lastly named group – learners who fill the interest – is excellent material to experiment with content of educational material and to try something new in teaching.

Based on this thesis results it could be concluded that there are no unreachable aims in educational process. Desire and motivation are needed, backed up by science and positive results.

REFERENCES

[1] Bonk, C. J. & Graham, C. R. Handbook of blended learning: Global Perspectives, local designs. San Francisco, CA: Pfeiffer Publishing.

[2] Wing M. J, Computational Thinking, Communications of the ACM, vol. 49(3), 33-35, 2006.

[3] Report of a Workshop on The Scope and Nature of Computational Thinking, USA National Academy of Sciences, 2010.

[4] Andrew Csizmadia, Paul Curzon, Mark Dorling, Simon Humphreys, Thomas Ng, Cynthia Selby, John Woollard. Computational Thinking - A guide for teachers. 2015 Available at: http://www.computingatschool.org.uk/computationalthinking

[5] Ying Li, Yu Liu, Pan Shu. Teaching Research and Practice of Blended Learning Model Based on Computational Thinking. Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE

[6] Miller, L. D. Improving learning of computational thinking using creative thinking exercises in CS-1 computer science courses. Frontiers in Education Conference, 2013 IEEE. 1426-1432.

[7] Linschner, R. Programming Language and Tools for Deep Learning (Elecrtonic Version). 2002. Available at: http://www.cs.utexas.edu/users/csed/doc_consortium/DC99/lischner-abstract.html

[8] Yuwanuch Gulatee, Barbara Combes. Identifying the Challenges in Teaching Computer Science Topics Online. Proceedings of the EDU-COM 2006 International Conference. Engagement and Empowerment: New Opportunities for Growth in Higher Education, Edith Cowan University, Perth Western Australia, 22-24 November 2006.

[9] Sue Sentence, Andrew Csizmadia. Computing in the curriculum: Challenges and strategies from a teacher's perspective. Education and Information Technologies. March 2017, Volume 22, Issue 2, 469–495.

[10] Challenges of Teaching GCSE Computer Science. Available at: https://teachcomputing.wordpress.com/2016/07/10/challenges-of-teaching-gcse-computer-science/

[11] MIT Media Lab, 2016. Scratch - Imagine, Program, Share. Available at: http://scratch.mit.edu/

[12] Maloney, J., et al. The scratch programming language and environment. Trans. Comput. Educ., 10(4):1–15, Nov. 2010.

[13] RESOURCES IN SCIENCE AND ENGINEERING EDUCATION. Richard Felder's Home Page. Available at: http://www4.ncsu.edu/unity/lockers/users/f/felder/public/

[14] CSTA K–12 Computer Science Standards. 2011. Available at: http://csta.acm.org/Curriculum/sub/K12Standards.html

[15] AP Computer Science Principles, 2011-2016. Available at: https://advancesinap.collegeboard.org/stem/computer-science-principles

[16] CSTA Computational Thinking Task Force. Available at: http://csta.acm.org/Curriculum/sub/CompThinking.html

[17] UK. The Royal Society. „Shut down or restart?“ The way forward for computing in UK schools. Available at: https://royalsociety.org/~/media/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf

[18] UK. Computing in the national curriculum: a guide for secondary teachers. Available at: http://www.computingatschool.org.uk/data/uploads/cas_secondary.pdf

[19] Computing: a curriculum for schools, Computing at School Working Group. Available at: http://www.computingatschool.org.uk, 2011

[20] Mark Guzdial. What’s the Best Way to Teach Computer Science to Beginners? Communications of the ACM. 2015, Vol. 58, no. 2. 12-13.

[21] Roels, R., Mestereaga, P. and Signer, B. An Interactive Source Code Visualisation Plug-in for the MindXpres Presentation Platform. Springer International Publishing Switzerland 2016. S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, 169–188, 2016.

[22] Lowther, J., What is Computer Science? Available at: https://www.cs.mtu.edu/~john/whatiscs.html

[23] Denning, P., Is Computer Science Science? Communications of the ACM, 2005, Vol. 48, No. 4, 27 – 31.

[24] Kinnunen, P., Challenges of Teaching and Studying Programming at a University of Technology – Viewpoints of Students, Teachers and the University. 2009. Doctoral Dissertation.

[25] Isabel C. Moura, Natascha van Hattum-Janssen. Teaching a CS introductory course: An active approach. Computers & Education. 56 (2011), 475–483.

[26] Hawi N., Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. Computers & Education. 54 (2010), 1127–1136.

[27] Shaobing Song, Ge Yu. Education Reform on Software Programming Courses for Non-IT-Majoring Undergraduates. The 6th International Conference on Computer Science & Education (ICCSE 2011).

[28] Veronica Dahl, Gemma Bel-Enguix, Diana Cukierman, M.Dolores Jiménez-Lopez. Logic Programming: Teaching Strategies for Students with no Programming Background. WCCCE '10.

[29] Bassey Isong, Ohaeri Ifeoma and Naison Gasela. On the Integration of Agile Practices into Teaching: an approach to overcoming teaching and learning challenges of programming. 2015 International Conference on Computational Science and Computational Intelligence. doi 10.1109/CSCI.2015.153

[30] Maurer, F. and Martel, S. "Extreme programming: Rapid development for web-based applications," IEEE Internet Computing., vol. 6, no. 1, 86–90, Jan./Feb.2002

[31] Manifesto for Agile Software Development. Available at: http://www.agilemanifesto.org

[32] ECDL Foundation, 2016. ECDL Foundation. Available at: http://www.ecdl.com

[33] Running On Empty: the Failure to Teach K–12 Computer Science in the Digital Age, Association for Computing Machinery (ACM), and Computer Science Teachers Association (CSTA). 2011. Available at: http://csta.acm.org

[34] Shut down or restart? The way forward for computing in UK schools, UK Royal Academy of Engineering, 2012

[35] Informaatika mitteinformaatikutele. Available at: https://docs.google.com/presentation/d/1skl44TVbctNU6ISBkFVmWqm34cDMxVU1cipsXExaQs0/edit#slide=id.p4

[36] International Society for Technology in Education and the Computer Science Teachers Association. (2011). Operational definition of computational thinking for K-12. Available at: http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

[37] Phillip Snalune. The benefits of computational thinking. ITNOW (Winter 2015) 57 (4): 58-59 doi:10.1093/itnow/bwv111. Available at:

http://itnow.oxfordjournals.org/content/57/4/58.full.pdf+html?sid=8d512e08-08ca-4c07-83fe-024176b512f1

[38] Geoffrey G. Roy, Joel Kelso, Craig Standing, "Towards a Visual Programming Environment for Software Development," Proceedings on Software Engineering: Education & Practice, 1998, 381-388.

[39] Min Hu, "A Case Study in Teaching Adult Students Computer Programming", Proceedings of the 16th Annual NACCQ, 2003, 287-291.

[40] Moodle, 2016. Open-source learning platform. Available at: https://moodle.org/

[41] ACM, IEEE Computer Society, Computer Science Curricula 2013: Curriculum Guides for Undergraduate Degree Programs in Computer Science. December 20, 2013.

[42] Study Information System, Tallinn University of Technology. Available at: http://ois.ttu.ee

[43] LEARNING STYLES. Available at: http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Learning_Styles.html

[44] Qingguo Zhou, Jiong Wu, Ting Wu, Jun Shen, Rui Zhuou. Learning Network Storage Curriculum With Experimental Case Based on Embedded Systems. 2015 Wiley Periodicals, Inc., 187-194.

[45] College Board, 2015. AP Computer Science Principles. Available at: https://advancesinap.collegeboard.org/stem/computer-science-principles

[46] Computer Science Teachers Association, 2015. Computational Thinking Task Force. Available at: http://csta.acm.org/Curriculum/sub/CompThinking.html

[47] Allaa Barefah, Elspeth McKay. Designing for Online Learning Environments: Towards an ePedagogy Development Model. 2015 IEEE Conference on e-Learning, e-Management and e-Services. 175-180.

[48] Khan Academy, 2015. Khan Academy. Available at: https://www.khanacademy.org/

[49] Broadbent, B., 2002. ABCs of e-Learning: Reaping the Benefits and Avoiding the Pitfalls. 1st ed. NY: John Wiley & Sons, Inc.

[50] Kris M.Y. Law, Victor C.S. Lee, Y.T. Yu, Learning motivation in e-learning facilitated computer programming courses. Computers & Education, 55 (2010), 218–228

[51] Bashinski, S. M., 2002. Adapting the Curriculum to Meet the Needs of Diverse Learners. PBS Teachers. Available at: http://www.pbs.org/teachers/earlychilhood/articles/adapting.html

[52] Kolb, D., 1984. Experiential Learning. Engle Cliffs: Prentice Hall.

[53] Dunn, R. (2003). The Dunn and Dunn learning style model and its theoretical cornerstone. In R. Dunn & S. Griggs (Eds.), Synthesis of the Dunn and Dunn learning styles model research: Who, what, when, where and so what (pp. 1–6). New York: St. John's University.

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

[55] Gregorc, A.F. (2015). Mind Styles & Gregorc Style Delineator. Available at: http://gregorc.com

[56] Brusilovsky, P. & Peylo, C. (2003). Adaptive and intelligent web-based educational systems. International Journal of Artificial Intelligence in Education, 13 (2–4, Special Issue on Adaptive and Intelligent Web-based Educational Systems), 159–172.

[57] Huong May Truong. Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities. Computers in Human Behavior 55 (2016), 1185–1193

[58] Sabine Graf, Tzu-Chien Liu, Kinshuk, Nian-Shing Chen, Stephen J.H. Yang. Learning styles and cognitive traits – Their relationship and its benefiits in web-based educational systems. Computers in Human Behavior 25 (2009) 1280–1289

[59] Ebru Özpolat, Gözde B. Akar. Automatic detection of learning styles for an e-learning system, Computers & Education 53 (2009) 355–367.

[60] Felder, R. M. and Spurlin, J. E., 2005. Applications, Reliability, and Validity of the Index of Learning Styles. Intl. Journal of Engineering Education, 21(1), 103-112.

[61] Felder, R.M. and Brent, R. Understanding Students Differences, Journal of Engineering Education, 2005, 94(1), 57-72. http://dx.doi.org/10.1002/j.2168-9830.2005.tb00829.x

[62] Amiera Syazreen Mohd Ghazali, Siti Fadzilah Mat Noor, Saidah Saad. Review of Personalized Learning Approaches and Methods in e-Learning Environment. The 5th International Conference on Electrical Engineering and Informatics 2015

[63] Aleksandra Klasnja-Milicevic, Boban Vesin, Mirjana Ivanovic, Zoran Budimac. E-Learning personalization based on hybrid recommendation strategy and learning style identification. Computers & Education, 56 (2011) 885–899.

[64] M. Sedleniece and S. Cakula. Framework for personalized elearning model, 457–462. doi:10.1016/j.procs.2013.12.011

[65] Cakula, S. and Sedleniece, M. Improvement of Personalized e-Learning Framework Using Principles of Knowledge Management. In: WSEAS, Recent Researches in Information Science and Application; 2013

[66] Amiera Syazreen Mohd Ghazali, Siti Fadzilah Mat Noor, Saidah Saad. Review of Personalized Learning Approaches and Methods in e-Learning Environment. The 5th International Conference on Electrical Engineering and Informatics 2015

[67] Keefe, J.W., "Learning Style: An Overview," in Keefe, J.W., ed., Student Learning Styles: Diagnosing and Prescribing Programs, Reston, Va.: National Association of Secondary School Principals, 1979.

[68] Felder, R.M and Soloman, B.A (n. d.) Learning styles and strategies. Available at: http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSdir/styles.htm

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

[70] Soloman, B. A. and Felder, R.M. (n. d.). Index of Learning Styles Questionnaire. Available at: http://www.engr.ncsu.edu/learningstyles/ilsweb.html

[71] Felder, R.M. and Spurlin, J. Applications, Reliability, and Validity of the Index of Learning Styles. Intl. Journal of Engineering Education, 2005, 21(1), 103-112. Available at: http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSdir/ILS_Validation%28IJEE%29.pdf

[72] Oakley, B., Felder, R., Brent, R., & Elhajj, I. (2004). Turning student groups into effective teams. Journal of Student Centered Learning, 2(1), 9–34.

[73] Toktarova, V., Panturova, A. Learning and Teaching Style Models in Pedagogical Design of Electronic Educational Environment of the University. Mediterranean Journal of Social Sciences. MCSER

Publishing, Rome-Italy. 2015. Vol 6 No 3 S7, 281-290. Doi:10.5901/mjss.2015.v6n3s7p281

[74] Student's t-Tests. Available at: http://www.physics.csbsju.edu/stats/t-test.html

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

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

[77] Sohail Iqbal Malik & Jo Coldwell-Neilson. A model for teaching an introductory programming course using ADRI. Education and information technologies. Springer Science+Business Media New York 2016. DOI 10.1007/s10639-016-9474-0

[78] Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and Teaching Programming: A Review and Discussion. Computer Science Education 2003, Vol. 13, No. 2, 137–172.

[79] Snap! Available at: https://snap.berkeley.edu/

[80] Blocky. Google Developers. Available at: https://developers.google.com/blockly/

[81] Jose-Manuel Saez-Lopez, Marcos Roman-Gonzalez, Esteban Vazquez-Cano. Visual programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools. Computers & Education 97 (2016), 129-141.

[82] Andrea Minuto, Fabio Pittarello, Anton Nijholt. Smart material interfaces for education. Journal of Visual Languages and Computing 31(2015), 267–274

[83] K. Brennan and M. Resnick, "Imagining, Creating, Playing, Sharing, Reflecting: How Online Community Supports Young People as Designers of Interactive Media," in Emerging Technologies for the Classroom, ser. Explorations in the Learning Sciences, Instructional Systems and Performance Technologies, C. Mouza and N. Lavigne, Eds. Springer New York, Jan. 2013, pp. 253-268.

[84] Orni Meerbaum-Salant, Michal Armoni, Mordechai (Moti) Ben-Ari. Learning computer science concepts with Scratch. http://dx.doi.org/10.1080/08993408.2013.832022

[85] Sayamindu Dasgupta, Shane M. Clements, Abdulrahman Y. idlbi, Chris Willis-Ford, and Mitchel Resnick. Extending Scratch: New

Pathways into Programming. 2015 IEEE Symposium on Visual Languages and Human-Centric Computing.

[86] M. Resnick, B. Silverman, Y. Kafai, J. Maloney, A. Monroy Hernandez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, and J. Silver, "Scratch: Programming for All," Communications of the ACM, vol. 52, p. 60, Nov. 2009.

[87] Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. American Educational Research Association Meeting. Vancouver, BC: Canada. Available at: http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

[88] Juha Sorva, Visual Program Simulation in Introductory Programming Education. Doctoral thesis. 2012

[89] TIOBE the software quality company. Available at: http://www.tiobe.com/tiobe-index/

[90] Randy Kaplan. Choosing a First Programming Language. SIGITE'10

[91] L. Grandell. "Why Complicate Things? Introducing Programming Language in High School using Phyton". Australasian Computing Education Conference, Vol. 52, 2006.

[92] Vambola Leping, Marina Lepp, Margus Niitsoo, Eno Tõnisson, Varmo Vene, Anne Villems. Python Prevails. International Conference on Computer Systems and Technologies - CompSysTech'09.

[93] Luiz Carlos Begosso, Luiz Ricardo Begosso, Emiliana Martins Gonçalves, Jean Rafael Gonçalves. An approach for teaching algorithms and computer programming using Greenfoot and Python. Proceedings - Frontiers in Education Conference, 2012

[94] Goldwasser, Michael H. and Letsche David. Using Python To Teach Object-Oriented Programming in CS1. In Innovation and Technology in Computer Science Education. Proceedings of The 13th Annual Conference on Innovation and Technology in Computer Science Education. Vol. 40. New York: ACM, 42-46.

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

[96] C. Vorderman, Computer Coding for Kids - A Unique Step-by-step Visual Guide, From Binary Code to Building Games. London, DK Children, 2014.

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

[98] Kanis Charntaweekhun, Somkiat Wangsiripitak. Visual Programming using Flowchart. ISCIT1 2006.

[99] Ball, S. (1977). Motivation in education. Academic Press.

[100] Entwistle, N. (1998). Motivation and approaches to learning: Motivating and conceptions of teaching. In Brown et al. (Eds.), Motivating students. Kogan Page.

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

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

[103] Jenkins, T. (2001). The motivation of students of programming. In Proceedings of ITiCSE 2001: The 6th annual conference on innovation and technology in computer science education. 53–56.

[104] Lawson, R. J. The Effect of Viva Assessment on Students' Approaches to Learning and Motivation. International Review of Social Sciences and Humanities. Vol. 2, No. 2 (2012), 120-132.

[105] About active learning. Available at: http://www.educationscotland.gov.uk/learningandteaching/approaches/activelearning/about/what.asp

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

[107] Brenda Cheanga, Andy Kurniaa, Andrew Limb, Wee-Chong Oonc. On automated grading of programming assignments in an academic institution. Computers & Education 41 (2003), 121–131.

[108] Roberto A. Bittencourt, David Moises B. dos Santos, Carlos A. Rodrigues, Washington P. Batista, Henderson S. Chalegre. Learning Programming with Peer Support, Games, Challenges and Scratch. doi:10.1109/FIE.2015.7344222

[109] Crede, M., Roch, S. G., and Kieszczynka, U. M. 2010. Class attendance in college: a meta-analytic review of the relationship of class attendance with grades and student characteristics. Review of Educational Research. 80, 2 (Jun. 2010), 272–295.

[110] MICHAL ARMONI, ORNI MEERBAUM-SALANT, and MORDECHAI BEN-ARI, From Scratch to "Real" Programming. ACM Transactions on Computing Education, Vol. 14, No. 4, Article 25, 2015.

[111] Exploring Computer Science. Available at: http://www.exploringcs.org/

[112] National Science Foundation. Retrieved from: http://www.nsf.gov/

[113] M. Resnick, J. Maloney, A. Monroy-Hernández, and others, Scratch: Programming for All, Communications of the ACM, vol. 52 (11), pp. 60-67, 2009

[114] Nathan Bean, Joshua Weese, Russell Feldhausen, R. Scott Bell. Starting From Scratch. Developing a Pre-Service Teacher Training Program in Computational Thinking. 2015. doi.ieeecomputersociety.org/10.1109/FIE.2015.7344237

[115] Patricia Haden, Joy Gasson, Krissi Wood, Dale Parsons. Can You Learn To Teach Programming in Two Days? ACE '16 Canberra, ACT Australia. http://dx.doi.org/10.1145/2843043.2843063

[116] Sue Sentance & Andrew Csizmadia. Computing in the curriculum: Challenges and strategies from a teacher's perspective. Educ Inf Technol (2016). doi:10.1007/s10639-016-9482-0

[117] Vitri Tundjungsari. E-Learning Model for Teaching Programming Language for Secondary School Students in Indonesia. 2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV). 262-266

[118] ScratchEd. Available at: http://scratched.gse.harvard.edu/

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

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

[121] Rakenduste loomine Scratchiga. Available at: http://rlpa.ttu.ee/scratch/Scratch_20/Scratch_20_P.html

[122] The Use of Scratch in Estonia. Available at: https://drive.google.com/drive/folders/0Bwxop-tjmpUXdXBjNGZueHdsUzg

[123] Scratch - Connecting Worlds. Available at: http://www.scratch2013bcn.org/

[124] Dr.Scratch. Available at: http://www.drscratch.org/

[125] J. Moreno-Leon and G. Robles. Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills. In Scratch Conference 2015, Amsterdam, The Netherlands, August 12-15, 2015, Proceedings, 48-53, 2015.

[126] J. Moreno-Leon and G. Robles. Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. DOI: http://dx.doi.org/10.1145/2818314.2818338

[127] David J. Malan, SC50. Available at: https://cs50.harvard.edu/

[128] A Computer Science Principles Course. Available at: https://cs.uteach.utexas.edu/

[129] Available CS Courses. Available at: http://apcsprinciples.org/

[130] The Beauty and Joy of Computing. Available at: http://bjc.berkeley.edu/

[131] Scratch - Teach Computer Programming in Schools. Available at: https://alison.com/courses/Scratch-Teach-Programming-in-Schools

[132] Introduction to Programming with Scratch in Education. Available at: https://uni-cs4hs-scratch.appspot.com/preview

[133] Code Yourself! An Introduction to Programming. Available at: https://www.coursera.org/learn/intro-programming

[134] Programming in Scratch. Available at: https://www.edx.org/course/programming-scratch-harveymuddx-cs002x-1

[135] CS For All: Introduction to Computer Science and Python Programming. Available at: https://www.edx.org/course/cs-all-introduction-computer-science-harveymuddx-cs005x-0

[136] Scratch Programming for Elementary School Students. Available at: http://cty.jhu.edu/ctyonline/courses/computer_science/scratch_programming_elementary.html

[137] Programming for Kids. Available at: https://www.udemy.com/programming-from-scratch/

[138] Adventures in Scratch Programming Course. Available at: https://www.idtech.com/kids/tech-camps/courses/adventures-in-programming-with-scratch/

[139] Scratch1: Getting started with Scratch. Available at: https://www.yorksj.ac.uk/ils/digitaltraining/bookable-courses/programming-courses/

[140] Scratchi materjalid. Available at: https://courses.cs.ut.ee/t/scratch

[141] ‘Scratch’. Available at: http://scratchime.weebly.com/

[142] Work done in Estonia for increasing society’s commitment in ICT. Available at: https://sisu.ut.ee/sites/default/files/ict/files/eno_tonisson.pdf

ACKNOWLEDGEMENTS

First, I would like to thank everyone who have advised and supported me during my PhD studies.

I would like to thank professor Rein Kuusik for providing guidance and encouragement in finishing this thesis.

I would like to thank my supervisors, Tiia Rüütmann and Enn Õunapuu for their guidance.

In addition, I would like to thank my Master's thesis supervisors – Irina Amitan and Jüri Vilipõld for their continuous motivation and support.

I am thankful students for participating in experiments.

Finally, I would like to thank my family for their great patience and support.

ABSTRACT

The goal of this doctoral thesis is to analyse the existing and develop new teaching technique for teaching Computer Science Basics to novices.

Present work describes teaching strategies and their combinations, which are used to compile and teach the Computer Science Basics course developed during last years, from 2010. This course is aimed to the first year non-IT students at Tallinn University of Technology and it acts as a model of present research.

The aim of developed strategies is to achieve better results in teaching and learning through the combination of Richard M. Felder learning style theory and students' prior knowledge level. The main research methodology of this thesis is based on the known principles of blended learning.

During the present research the teaching approach to the Computer Science basics was studied, reviewed and developed. Moreover, the content of teaching materials was greatly renewed in accordance with the results of the study.

To study the teaching approach and achieve the research goal various tests were carried out to identify students' levels of knowledge and learning preferences. Throughout the long-term experiment, the first year non-IT students, who are beginners in Computer Science field, were divided into groups according to tests outcomes. These groups were formed of students with different levels of prior knowledge and learning styles. During their study students have got learning materials that were developed based on knowledge, which course instructors have got from tests. The chosen teaching approach comprises a combination of successful strategies, such as e-support and students' background study, to enhance their positive effect.

Furthermore, the author of this work considers, analyses and develops the main techniques and principles of programming teaching for the beginners in this field. There under the beginners the author of the thesis understands non-IT students, school pupils and schoolteachers who are starting to learn programming basics.

The first important result of this research is new combination of known teaching strategies – worked out course teaching model, which can be successfully applied for the teaching in other Computer Science courses. The success of chosen strategy is demonstrated by comparing the achievements of the *test* groups, who were taught using the developed strategies, with the *reference* groups, who were taught in a conventional manner. Renewed and supplemented learning materials were provided for the students considering the two main aspects: the level of their prior knowledge and their preferred learning style.

The second achievement of this thesis is the programming foundations teaching methodology, which is developed and applied in practice in the Computer Science Basics course as well as in additional courses in Tallinn University of Technology. The main idea of this approach is model-based programming teaching that is founded on the visual programming, which is integrated into courses before any serious coding. The goal of this methodology is to make clear main principles of modelling and algorithmization and accordingly give students the knowledge base for exploring the textual coding.

The Computer Science Basics course for the first year non-IT students from economics, social, chemistry and civil engineering faculties at Tallinn University of Technology and courses for school pupils and schoolteachers are as a model of teaching and learning using this strategy. Despite the fact that TUT matriculant's level of prior IT-knowledge constantly decreasing the author of this thesis has worked out the new teaching model, which provides support for students with appropriate learning materials, and help them to achieve high level of knowledge.

Combining abovementioned results it could be claimed that the main contribution of the undertaken work is new successful teaching strategy for teaching Computer Science Basics to novices.

KOKKUVÕTE

Antud doktoritöö eesmärk on analüüsida olemasolevat, ning arendada uut õppetehnikat üldinformaatika õpetamiseks algajatele.

See uurimistöö kirjeldab õpetamisstrateegiat, mida on kasutatud viimastel aastatel – alates aastast 2010 – välja töötatud üldinformaatika kursuse arendamisel ja õpetamisel algajatele. Kursus on suunatud Tallinna Tehnikaülikooli esimese aasta mitte IT eriala üliõpilastele.

Esitatud õpetamisstrateegia põhieesmärk on parimate tulemuste saavutamine õppimises ja õpetamises, kombineerides Richard M. Felderi õppestiilide teooriat ning tudengite eelteadmiste taset. Kasutatud metodoloogia baseerub segaõppe tuntud printsiipidel.

Doktoritöö käigul oli uuritud, analüüsitud ja arendatud õpetamislähenemine Informaatika algkursusel. Pealegi kursuse sisu ja materjalid olid uuendatud vastavalt töö tulemustele.

Õppurite teadmiste taseme määramisel ja nende õppimisstiilil põhineva õppeprotsessi eelistuste tuvastamiseks viidi läbi suur hulk erinevaid teste. Selles pikaajalises eksperimendis jagati esimese aasta mitte IT-eriala üliõpilased vastavalt testi tulemustele rühmadesse. Seega moodustati grupid, mis vastasid õppurite eelnevatele teadmiste (eelteadmistele) tasemetele ja õpistiilidele vastavalt testide tulemustele. Valitud lähenemine sisaldab selliste edukate strateegiate nagu e-tugi ja õppurite õpistiili taustauuring ühendamist nende positiivse koosmõju suurendamiseks.

Peale selle käesoleva töö autor vaatleb, analüüsib ning arenda programmeerimise õpetamise põhiprintsiipe algajatele selles valdkonnas. Algajatena määratleb töö autor mitte IT-eriala üliõpilasi, kooliõpilasi ja õpetajaid, kes hakkavad õppima programmeerimise aluseid.

Doktoritöö esimeseks tulemuseks on uus õppestrateegiate kombinatsioon - valmis õpetamismudel, mida saab kasutada teiste arvuti kursuste õpetamisel. Strateegia edu demonstreerib testirühmade tulemuste tasemete võrdlemine – paremaid tulemusi saavutasid need, kelle õpetamisel kasutati väljatöötatud strateegiaid võrreldes nendega, keda õpetati tavapärasel viisil.

Teiseks doktoritöö tulemuseks on väljatöötatud strateegia programmeerimise õpetamiseks, mida on rakendatud praktikasse Tallinna Tehnikaülikoolis nii informaatika õpetamise kursustel kui ka täiendusõppe kursustel. Selle lähenemise peamiseks ideeks on programmeerimise õpetamine mudeli põhjal, mis põhineb visuaalsel programmeerimisel ja on integreeritud kursusesse enne tõsist programmeerimiskoodi kirjutamist, võimaldades selle läbi eelnevalt lihtsalt ja loomulikult selgeks teha modelleerimise ja algoritmimise

põhiprintsiibid, mis tagavad teadmiste baasi erinevate programmeerimiskeelte õppimiseks.

Innovaatiline informaatika kursus, milles uuendatud ja täiendatud õppematerjalid arvestavad kahte peamist aspekti, eelteadmiste taset ja eelistatud õpistiili, on mudelina läbiviidud uuringus. Kursus on suunatud majandus-, sotsiaal-, keemia- ja ehitusteaduskonna tudengitele. Hoolimata sellest, et TTÜsse sisseastujate IT alaste teadmiste tase on pidevalt halvenenud, on autor suutnud välja töötada õpetamismudeli, mis toetab õppureid asjakohaste õppematerjalidega ja aitab neil saavutada kõrget teadmiste taset.

Kokkuvõttes saab öelda, et doktoritöö põhitulemus on uus edukas õpetamistehnika üldinformaatika õpetamisel algajatel

Appendix A

Mironova, O.; Rüütmann, T.; Amitan, I.; Vilipõld, J.; Saar, M. (2013). Computer Science E-Courses for Students with Different Learning Styles. In: Annals of Computer Science and Information Systems, 1: Federated Conference on Computer Science and Information Systems, September 8–11, 2013. Kraków, Poland. IEEE, 735−738.

Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 735–738

Computer Science E-Courses for Students with Different Learning Styles

Olga Mironova, Irina Amitan,
Jüri Vilipõld, Merike Saar
Faculty of Information Technology, Department of Informatics, Chair of Software Engineering, Tallinn University of Technology, Akadeemia tee St. 15A, Tallinn 12618, Estonia
Email: {olga.mironova, irina.amitan, juri.vilipold, merike.saar}@ttu.ee}

Tiia Rüütmann
Faculty of Social Sciences, Department of Industrial Psychology, Estonian Centre for Engineering Pedagogy, Tallinn University of Technology, Akadeemia tee St. 3, Tallinn 12618, Estonia
Email: tiia.ruutmann@ttu.ee

***Abstract*—E-learning is a contemporary teaching tool that has become popular and widely used in engineering education in recent years. This article presents the outcomes of a study on considering students` different learning styles in teaching information and communication technology using e-learning. Students were divided into two study groups. The reference group studied according to a provided learning model which including both theoretical educational material and practical assignments. Students of the test group were divided according to their learning styles using the Felder-Silverman model. Different relevant learning models, which included the same theoretical material and practical assignments, were designed for students of the test group based on the learning styles. The results of the study proved that the learning materials which were designed taking into account students` different learning styles considerably improved the achievement of the learning outcomes. A detailed description and analysis of the study is presented in the article.**

I. Introduction

ENGINEERING education is a large system and it is almost impossible to predict its behaviour over far too distant future since the system parameters show a high rate of change. All knowledge is changing so fast that we cannot give students what they will need to know tomorrow. Instead, we should be helping them develop their learning skills so that they will be able to learn whatever they need to. If we can achieve that, we will have world-class engineers, people who are innovative and resourceful.

Learning styles are characteristic cognitive, affective, and psychological behaviours that serve as relatively stable indicators of how learners perceive, interact with, and respond to the learning environment. Students learn best when instruction and learning context match their learning style.

Understanding students' different learning styles is one of the midpoints of effective education. The aim of the research described in the article was to abolish mismatches between students' common learning styles and teaching styles in e-learning and make teaching in engineering more effective.

According to Felder and Brent [1], students learn in many ways – by seeing and hearing; reflecting and acting; reasoning logically and intuitively; memorising and visualising; drawing analogies and building mathematical models.

Classroom activities of teachers and students take place in mutual communication. Therefore, the guidance and the formative role of the teacher should be realized in the creation and review of theoretical material and the material in practical classes. However, most of the learning processes are individual learning activities and here self-regulation of the student is realised. The task of the teacher in this case is to provide students with a supportive learning environment: motivate, guide and support. It should be noted that learning should be based on individual personality traits. This ensures successful acquisition of knowledge.

II. Methodology

Since 2010, we have applied a flexible, adaptive approach to teaching computer science in Tallinn University of Technology. The main idea of this method was students division into groups according to their prior subject knowledge. The tasks were also of different level and it has given visible results – the level of knowledge has increased [6]. In teaching we have been focused our attention on activating an individual student's learning.

Students learn in different ways: some like to listen to and talk, while the others prefer to read texts or study by investigating the charts, diagrams and drawings. Any learning style can give good results if it is timely identified and a right approach is chosen and applied.

Teaching must transfer knowledge and support learning, but it must also be cooperative and directed toward students' reflection and development. Helping students in finding and forming their own style of learning – should customize the learning process aimed at creating the conditions for each student for the maximum development of his/her abilities, aptitudes, satisfaction of cognitive needs and interests.

Since the beginning of the fall semester 2012, we have conducted experiments in which we have tried to identify the most suitable learning activities for students, based on an individual test on learning styles. 300 students of economics, social and technical disciplines have been involved in the experiment. In the e-environment Moodle (https://moodle.e-ope.ee/) students were divided into two equal groups of 150 participants: a reference group and a test group.

978-1-4673-4471-5/$25.00

Students of both groups were taught the informatics courses depending on their prior knowledge: a test was carried out dividing them into beginners, advanced, and experts users. For beginners – the test result was 0% – 60%; for advanced – the test result was 61% – 80%, for experts – it was 81% – 100%. The test contained a different number of computer science related tasks with different difficulty levels.

The system and its effectiveness have been described in the article about a flexible approach to learning [6].

In addition, for the students of the test group all course materials and the whole learning process was designed to match their learning style preferences identified in the test [7].

Felder divides students based on their perception of the material and work with it into the following groups [2]:

active (ACT) and reflective (REF)
sensing (SEN) and intuitive (INT)
visual (VIS) and verbal (VRB)
sequential (SEQ) and global (GLO)

Active learners acquire new knowledge best by doing, discussing and explaining it to others in a group. At the same time reflective learners first think about it alone.

Sensing learners like learning facts and solving problems by well-known methods. Intuitive learners prefer discovering new possibilities and relationships and they are more innovative.

Visual learners remember pictures, diagrams, charts and video best. Verbal learners prefer written and spoken explanations.

Sequential learners like step by step studying, where each step follows logically from the previous one. Global learners prefer to get information by large portions and randomly.

The preferences of students, based on tests carried out among the students of the test group, are shown in Table I. The total for each student is 400% as each student could account for four different forms of information acquisition.

Data from Table I is shown in the following diagrams.

Tests carried out have shown that the majority of students do not have any preferences in the selection of learning materials and that they use a combination of different learning styles– they are well balanced (Fig. 1).

Figure 2 shows the types of students who acquire material better if their learning style has been taken into account. So, this group of students learns better if they are given possibilities to participate in group work, discuss, solve real tasks based on facts, etc.

However, some students have very strong preferences in learning. As presented in Figure 3, according to the tests active learners, sensing learners and visual learners fall into this group [4].

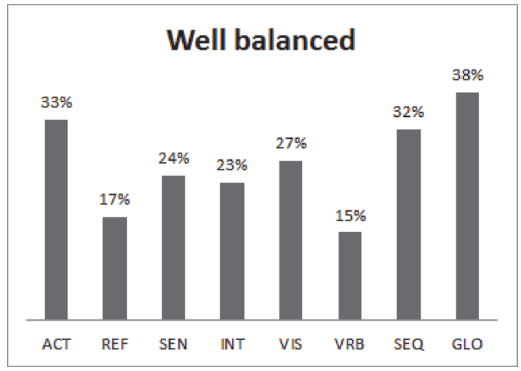

Fig. 1. Well balanced students

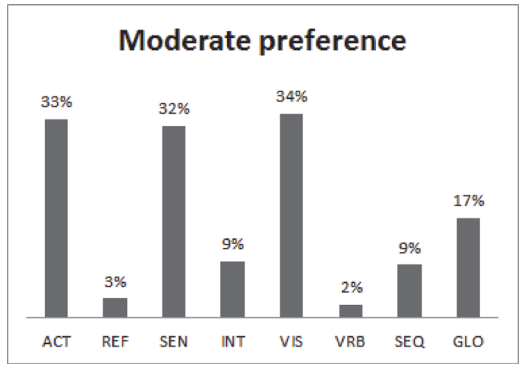

Fig. 2. Moderate preference

This way it was possible to find out the main preferences of students in the test group.

Based on the recommendations for the selection of educational material [3], [2], we designed and offered students assignments and theoretical materials according to their learning styles in the Moodle e-environment.

For example, to active learners we proposed group work assignments, to sensing learners – exercises, which were connected with solving real problems, and to visual learners – visual representation of course material, the same principles as have been used in the Khan Academy [5].

TABLE I.
THE PREFERENCES OF STUDENTS, OF THE TEST GROUP

	ACT	REF	SEN	INT	VIS	VRB
Well balanced	33%	17%	24%	23%	27%	15%
Moderate preference	33%	3%	32%	9%	34%	2%
Strong preference	13%	1%	10%	2%	22%	0%

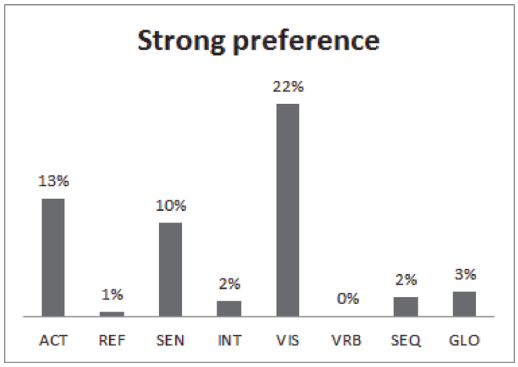

Fig. 3. Strong preference

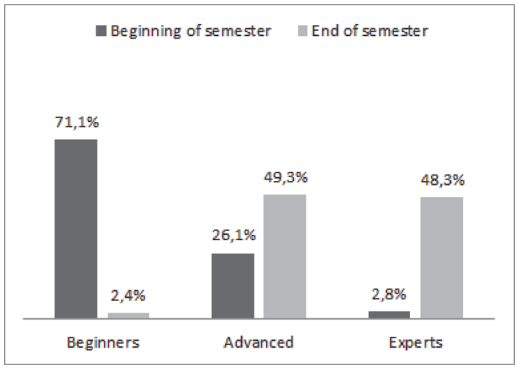

Fig. 4. Division of students into groups by test results in the reference group

All things considered, we managed to make the learning process more flexible by using e-environment opportunities: the students themselves chose the learning tempo, types of educational materials, and direction of individual and group work.

III. Results of the Experiment

The first results of our work showed a positive trend in the acquisition of knowledge. To divide students into groups by prior knowledge all of them were tested at the beginning of fall semester 2012. The same test was held at the end of fall semester. The test results confirm that students of the test group had mastered the learning material better than the students of the reference group (Fig 4 and 5).

Students of the test group coped better with their final exam due to the adopted learning material. Growth of knowledge has had a positive effect on their academic achievement (Fig 6).

Students' feedback in the test group also indicated that the material selected according to their learning styles motivated and helped them to learn.

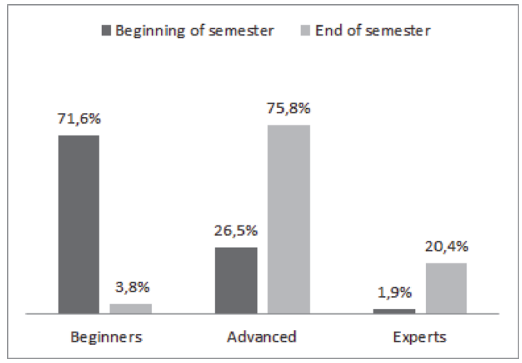

Fig. 5. Division of students into groups by test results in the test group

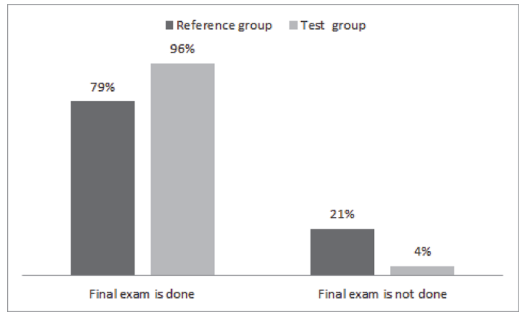

Fig. 6. Academic achievements. End of fall semester 2012

IV. Conclusions and Further Development

Students have different levels of motivation, different attitudes about teaching and learning, and different responses to specific classroom environments and e-learning. The more thoroughly teachers understand the differences, the better chance they have of meeting the diverse learning needs of all of their students. Teachers should attempt to improve the quality and efficiency of their teaching, which in turn requires understanding of the learning styles of students and designing instruction to meet these preferences.

Our selected flexible adaptive learning approach improved the quality of educational material and enhanced the educational effect of the use of innovative methods. The approach also provided us with additional opportunities to build individual educational paths for students, and in addition, apply the approach on students with different levels of readiness to learn.

Thus, we gave students the opportunity to choose their own way of learning the course. Students themselves felt the need for further studies, and did not feel the pressure from the teacher. They had the opportunity to work with educational materials in the manner and volume that was appropriate for them directly.

In conclusion, we would like to emphasize again that the content of the material adapted for each learning style should also cater for individualization of learning. It is important to remember that any learning style works well with the right approach.

Our chosen direction is a deeper study and analysis of students' data which could give us a better overview of why and how students learn. Additionally, there is the need for the curricula adaptation and teaching materials composition in accordance.

REFERENCES

[1] Felder, R.M. and Brent, R. Understanding Students Differences, Journal of Engineering Education, 2005, 94(1), pp. 57-72.

[2] Felder, R.M. and Silverman L.K. Learning and Teaching Styles in Engineering Education, Engr. Education, 1988, 78(7).

[3] Felder, R.M and Soloman, B.A (n. d.) Learning styles and strategies. Retrieved August 20, 2012, from http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSdir/styles.htm

[4] Felder, R.M. and Spurlin, J. Applications, Reliability, and Validity of the Index of Learning Styles. Intl. Journal of Engineering Education, 2005, 21(1), pp. 103-112.

[5] Khan Academy. http://www.khanacademy.org

[6] Mironova, O., Amitan, I., and Vilipõld, J. Computational Thinking and Flexible Learning: Experience of Tallinn University of Technology. Lecture Notes in Information Technology; 2012, 23-24, pp. 183 – 188.

[7] Soloman, B. A. and Felder, R.M. (n. d.). Index of Learning Styles Questionnaire. http://www.engr.ncsu.edu/learningstyles/ilsweb.html

Appendix B

Mironova, O.; Amitan, I.; Vendelin, J.; Saar, M.; Rüütmann, T. (2014). Strategies for the Individualization of an Informatics Course. Annals of Computer Science and Information Systems, 2: Federated Conference on Computer Science and Information Systems, September 7–10, 2014. Warsaw, Poland. IEEE, 835−840.

Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 835–840
DOI: 10.15439/2014F259
ACSIS, Vol. 2

Strategies for the Individualization of an Informatics Course

Olga Mironova, Irina Amitan, Jelena Vendelin, Merike Saar
Faculty of Information Technology, Department of Informatics, Chair of Software Engineering, Tallinn University of Technology, Akadeemia tee St. 15A, Tallinn 12618, Estonia
Email: {olga.mironova, irina.amitan, jelena.vendelin, merike.saar}@ttu.ee}

Tiia Rüütmann
Faculty of Social Sciences, Department of Industrial Psychology, Estonian Centre for Engineering Pedagogy, Tallinn University of Technology, Akadeemia tee St. 3, Tallinn 12618, Estonia
Email: tiia.ruutmann@ttu.ee

Abstract—The present paper describes the strategies used to compile and teach an Informatics course developed during last years at Tallinn University of Technology. The strategy is based on the main principles of blended learning and the analysis of the results of experiments with students from different faculties. Various tests were carried out to identify students' levels of knowledge and preferences in their learning process based on their learning styles. Throughout the experiment, students were divided into groups according to the test outcomes. Separate groups were formed of students with different levels of knowledge and learning styles, determined using the Felder-Silverman model.

Adaptive learning tools were provided for the students considering the three main aspects: students' background, the level of their prior knowledge and their preferred learning style. The success of the strategy presented in this article is demonstrated by comparing the achievements of the test group with the reference group, who were not taught using the new strategies.

I. Introduction

E-learning is a rapidly developing world-wide system. Currently, it is not possible to imagine any educational process without e-components or a holistic e-learning system. The main aim of such systems is to provide knowledge in a convenient form for its consumer – the learner. Abundance of information does not guarantee perfect knowledge. Teaching materials should be carefully structured to cater for the needs and preferences of the students.

All present-day knowledge in engineering education is changing so fast that we cannot predict what the 21th century students will need to know tomorrow. Instead, we should be helping them to develop learning skills and strategies so that they will be able to learn whatever they need to. A combined set of knowledge, skills and attitudes is essential to strengthen productivity, entrepreneurship and excellence in an environment which is based on technologically complex and sustainable products, processes and systems. Similarly, we could improve the quality and nature of engineering education. Thus the objective of engineering education today is to educate students who are ready to engineer, and deeply knowledgeable about technical fundamentals.

Computer science is an integral part of the curriculum, which contents change fast. The general aim of this course is to develop logical, analytical and computational thinking by using the computer on the highest level.

Considering the target audience, several attempts were made to design the course material in the way that it would be easy to understand but would still achieve the goals. Nevertheless, the course seemed to be rather difficult for most of the students. It resulted in low examination grades and lack of motivation.

It became clear that more adaptive learning tools and taking into account individual properties of each student would motivate them and, as a result, would lead to better academic achievements. The question remained how to achieve as much individualization of teaching as possible, using the existing time and personnel resources.

II. A Brief Description Of The Course

The Informatics course belongs to the curriculum of the Institute of Informatics at Tallinn University of Technology. The aim of the course, designed for the first year non-IT students, is creation of applications by using standard PC equipment and developing object-oriented computational thinking. The learning process starts with processing information using Excel spreadsheets: formulas, diagrams, built-in functions and facilities. The set of practical assignments depends on the students' specialization: economics, social, chemistry and civil engineering.

Further, students learn the basics of programming in practice and the main principles of algorithmization. Python for technical disciplines and Visual Basic for Application (VBA) for humanitarians have been picked out as the programming languages for the second part of this course.

It should be noted that the programming part of the course was complicated for most of the students, especially for the humanitarians. This issue was solved by implementing Scratch in the course curriculum. This intuitive graphical

978-83-60810-58-3/$25.00

programming language helps students to take on board the main ideas such as brunching and cycle.

The Informatics course lasts for two semesters. During this period, we try to combine different styles of teaching and learning: classic face-to-face classroom methods, group work and learning in the Moodle e-environment [9]. The last one gives us a huge amount of different opportunities for individualizing the learning process, such as adjustment of the learning pace, for example, as well as increase and variety in the number of learning assignments. Furthermore, students get a diversity of ways to learn and possibilities for self-tests in the e-part of the course. During the study time, they can choose between different kinds of teaching materials and use what they prefer based on their knowledge and learning styles.

The course is taught in three languages: Estonian, English and Russian.

III. Research Methodology

A. Overview

It is generally known that it is not possible to provide all students with one-to-one tutoring at a university. However, this fact should not affect the main educational goal - to ensure high-quality competitive knowledge. In our experiments with course design and curriculum, we considered the differences in students' features, especially their learning styles and prior knowledge.

Since the 2010 fall semester, our group of lecturers has applied a new flexible and adaptive approach to designing computer science e-courses [8]. The basic idea of our methodology is to divide students into equal reference and test groups and to compare the results of these two groups. The reference group is taught using the same course materials and the same e-course but they were not helped with any additional system. The students of the test group, however, were directed in choosing their learning materials based on the data obtained through the tests.

The division into groups was random and was not linked to the students' specialization or knowledge. During the 4 years of the experiment, the group sizes varied depending on the number of students. An average number of participants in the experiment each year was about 100-150 students and they were not aware of the research.

To evaluate the students' progress we tested them at the beginning of the course and at the end of it. The test contents for both groups were similar and were based on the topics described in the European Computer Driving License (ECDL) [1]. The tasks focused on creating documents and presentations, processing spreadsheets, and elementary knowledge in programming.

The last category of the tasks was added recently. In this implementation, we proceeded from an elective course 'Basics of Application Development and Programming', which has been included in the curriculum of Estonian secondary schools.

The current situation of teaching computer science at schools is varied. Some schools do not have computer science lessons at all due to the lack of teachers. Unfortunately, most of those pupils who have obtained sufficient IT-knowledge at their schools are not non-IT students at university level, who our course is aimed at and designed.

B. Research stages

Theoretically, it is possible to name three main strategies in the process of improving the Informatics course and individualization of the learning process:

- e-course
- knowledge
- learning style.

The first stage, which was named 'e-course', includes the adaptation of the course materials for the e-environment. Theoretical materials and practical assignments were innovated and supported with videos [6] and self-tests. They made the Informatics course more attractive for students. Both groups, the test and the reference group got access to this renewed course. At the same time, face-to-face lessons were held, too. Here we preferred group work that gave the students an opportunity to try the obtained knowledge in practice and develop teamwork skills. In this case the role of the lecturer was slightly different – it became more of an advisor, motivator and supporter in the student's work with learning materials. During the contact lessons learners ask questions related to their homework and share their skills and experience with other. In addition, students have an opportunity to get the support not only from their teacher but from other students, too. This form of support is equally useful both for the students who get it and, especially, for the ones who give it. To find and correct a mistake is an important skill in computer science subjects.

It makes no sense now to enumerate all the advantages of e-learning – they have been known to all. At this stage, we got the first results of our work: positive feedback from students and increase in academic achievements. These results were extracted from the Studies Information System (ÕIS) – an e-environment, where students and teachers get information about courses and curricula, students declare courses, keep results and give anonymous feedback on their educational process [11].

The second stage, titled 'knowledge', is dedicated to the division of the test group students into three streams based on their subject knowledge at the beginning of the course. Those three groups receive different amounts of different level practical tasks in the Moodle e-environment (Fig. 1). So, we increased the number of practical tasks without increasing the subject hours.

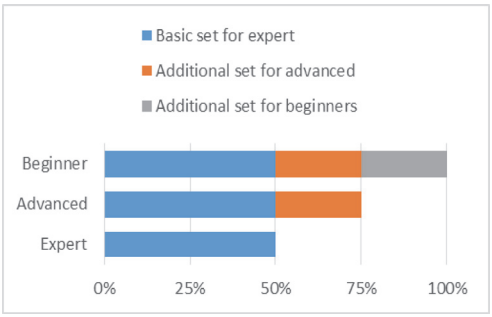

Fig. 1 The distribution of practical tasks

The face-to-face lessons were held as before. The mentioned division provided us with important information about what the students knew before studying our course. Moreover, according to the data we were able to provide them with the necessary learning material.

The success of this stage of experiment was confirmed by positive results of the students' survey in the ÕIS and by the increase in academic results.

The 'learning style' phase of our experiment was the most laborious part. Learning styles are characteristic cognitive, affective, and psychological behaviours that serve as relatively stable indicators of how learners perceive, interact with, and respond to the learning environment. Students learn best when instruction and learning context match their learning style.

There are many studies about the individualization of learning depending on students' ability [7], [3]. Using one of these, the learning styles model of Felder-Silverman, we divided our test group students by their preferences and provided them with corresponding learning materials [3], [4].

Felder distinguishes the following groups of learners depending on their learning styles [2]:

- active and reflective
- sensing and intuitive
- visual and verbal
- sequential and global.

Through this division, we found that the majority of our students were active and visual learners and they had very strong preferences for their learning process. These preferences were detected according to the Felder test which was held at the beginning of the course [10]. Throughout the educational process, students were provided with the necessary learning materials and activities in accordance with Felder´s instructions [5]. For example, active learners received more group work and opportunities to help others; visual learners were provided with visual representation of the educational material.

It should be noted that each year the number of active and especially visual students increased (Fig.2).

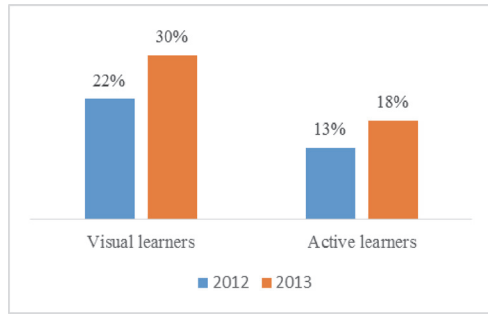

Fig. 2 Increase in the number of visual and active learners

The results of this stage of research showed us that the test group students managed with their practical tasks better than the students of the reference group. This has led to the better academic progress of the test group students.

Thus, we were able to create a model of our students´ learning preferences (Fig. 3), which considers their level of knowledge and preferences in the learning process. Using this model, we try to find an individual approach to each student in our course.

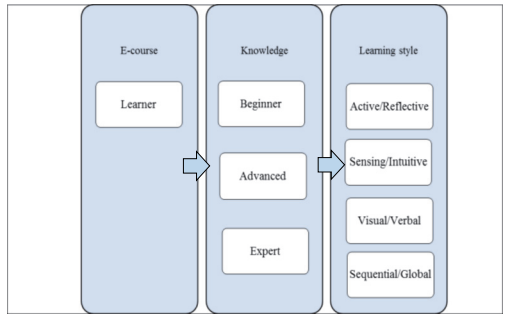

Fig. 3 The model of students' learning preferences

C. The Method of Research

Throughout the experiment, positive students feedback and good exam results showed the positive effect of the course and curriculum modifications. Finally, it was decided to examine the data with statistical methods. The aim of this examination is to check and prove the correctness of the chosen approach to an educational process.

As the method of the hypothesis testing, we chose the Student's t-test for the comparison of the two means. This test assumes a normal distribution of samples and not significant differences between the standard deviations of either samples. Our aim is to show that there were no significant differences between the test and reference group students in September, while in January the results are significantly different.

For calculations we use the equations for the averages $\bar{x}$ (1) and corresponding standard deviation S for both groups (2):

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad (1)$$

$$S = \sqrt{\frac{\sum_{i=1}^{n} \left(\bar{x} - x_i\right)^2}{n-1}} \quad (2)$$

After that, we calculated the standard error σ using the equation 3.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left(\bar{x}_{Test} - x_{iTest}\right)^2 + \sum_{i=1}^{n} \left(\bar{x}_{Ref} - x_{iRef}\right)^2}{n \cdot (n-1)}} \quad (3)$$

Finally, using equation 4 we calculated the experimental value t_{exp}:

$$t_{exp} = \frac{\left|\bar{x}_{Test} - \bar{x}_{Ref}\right|}{\sigma} \quad (4)$$

In addition, to compare this value with theory we need to calculate the degree of freedom df using equation 5:

$$df = 2n - 2 \quad (5)$$

As initial data for calculations, we chose September 2013 students' test results of the test and reference groups, and the same groups' results in January 2014 (cf. Table I in the Appendix).

Both samples are in the equal size: n=89 and distributed normally (Fig. 4 and Fig. 5).

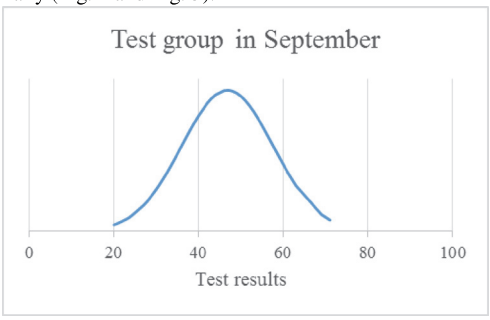

Fig. 4 The distribution of the test group results

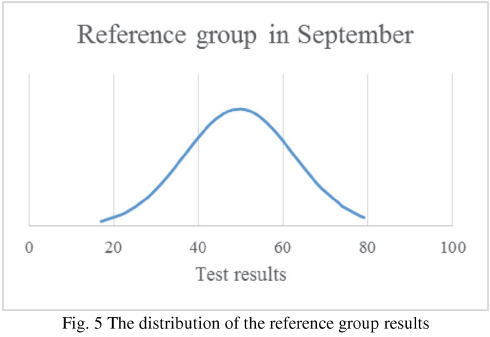

Fig. 5 The distribution of the reference group results

The two means and the corresponding standard deviations are calculated by using the equations 1 and 2:

$$\bar{x}_{Test} = \frac{4164}{89} \approx 46{,}787$$

$$\bar{x}_{Reference} = \frac{4413}{89} \approx 49{,}584$$

$$S_{Test} = \sqrt{\frac{10222{,}94}{89-1}} \approx 10{,}778$$

$$S_{Reference} = \sqrt{\frac{14721{,}62}{89-1}} \approx 12{,}934$$

Now we see that there is no significant difference between the standard deviations in either groups. It means that we can continue with Student tests.

The standard error of the difference between the two means is calculated by using equation 3:

$$\sigma = \sqrt{\frac{10222{,}94 + 14721{,}62}{89 \cdot (89-1)}} \approx 1{,}785$$

Experimental t value is calculated using equation 4:

$$t_{expSept} = \frac{|46{,}787 - 49{,}584|}{1{,}785} \approx 1{,}567$$

To compare this value with the theoretical t_{th} we need to calculate the degree of freedom using equation 5:

$$df = 2 \cdot 89 - 2 = 176$$

Using equations 1 to 4 we then calculated both groups´ results in January 2014:

$$\bar{x}_{Test} = \frac{8029}{89} \approx 90{,}213$$

$$\bar{x}_{Reference} = \frac{6896}{89} \approx 77{,}443$$

$$S_{Test} = \sqrt{\frac{5092{,}94}{89-1}} \approx 7{,}608$$

$$S_{Reference} = \sqrt{\frac{12518{,}22}{89-1}} \approx 11{,}927$$

$$\sigma = \sqrt{\frac{5092{,}94 + 12518{,}22}{89 \cdot (89-1)}} \approx 1{,}499$$

$$t_{expJan} = \frac{|90{,}213 - 77{,}443|}{1{,}499} \approx 8{,}489$$

D. Results of the experiment

Using the table of theoretical t_{th} values with the corresponding degree of freedom (cf. Table II in the Appendix) we found that the means of September results are not different at any critical level:

$$t_{expSept} < t_{th}$$

$$1{,}567 < 1{,}65$$

This means that at the beginning of the course both groups of students, the test and reference, had the same level of knowledge.

January results are the opposite – the means are different at critical levels.

$$t_{expJan} > t_{th}$$

$$8{,}489 > 3{,}29$$

It shows that the students who were taught using our system of the learning process individualization obtained knowledge much better then the others.

The progress of both groups is shown in the Figure 6 and Figure 7:

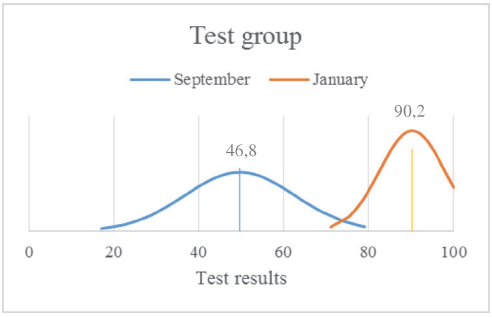

Fig. 6 The test group progress

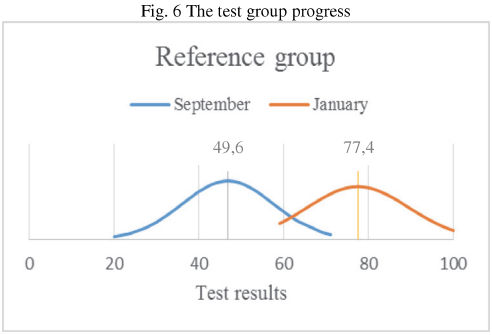

Fig. 7 The reference group progress

These results confirm the validity of our study and the chosen method of course individualization.

IV. Conclusions

The analysis presented in the paper shows positive outcomes of the strategy used. The calculations with two relevant groups, the test and reference, demonstrate significant differences in achievements at the end of the first part of the course.

The feedback, received from the groups, is also different. The test group shows higher motivation for further learning compared to the reference group. The main reason for it, picked out by students, was that there were no unreachable targets in the educational process.

Classroom activities of teachers and students took place in mutual communication. Therefore, the guidance and the formative role of the teacher was realized in the creation and review of the theoretical material and the material in practical classes.

The authors intend to continue developing the created model and Informatics course in the same style, trying to adapt it to individual students as much as possible.

Appendix

The results of each group are arranged in two columns under the name of the group in the Table I.

Table II shows only a part of the table of theoretical t values for Student's test. The whole table could be found in any book on statistical analysis.

TABLE I.
THE TEST RESULTS OF BOTH GROUPS

September 2013				January 2014			
Groups' results				Groups' results			
Test		Reference		Test		Reference	
63	52	61	39	89	91	79	72
52	40	54	59	81	98	84	100
71	38	78	48	75	91	78	71
63	40	59	46	91	79	80	62
53	56	55	40	100	96	77	74
69	58	58	44	79	98	81	68
61	48	69	65	71	78	64	60
69	54	74	57	83	98	60	72
61	55	60	47	100	100	67	87
63	40	56	50	89	81	71	86
61	51	56	64	88	92	84	100
55	42	61	48	79	98	60	84
42	39	51	42	89	80	100	78
59	44	59	46	87	90	79	82
63	48	53	48	100	90	73	84
54	55	60	54	89	98	100	92
48	47	44	46	77	90	66	84
49	40	55	54	88	93	66	92
55	32	63	44	85	95	78	82
44	54	54	17	90	98	96	76
52	47	61	55	88	86	78	94
57	44	79	51	100	87	66	60
55	46	74	50	81	82	72	59
46	44	53	25	77	91	72	65
59	34	65	23	90	98	84	63
55	35	59	30	86	82	78	70
48	28	36	33	79	98	67	74
46	40	43	59	87	89	75	100
53	37	59	55	98	98	91	98
44	43	55	25	76	96	99	69
47	43	46	52	90	90	69	96
52	30	69	34	100	98	62	78
56	36	68	36	80	94	78	82
48	44	55	50	92	92	79	96
44	34	46	41	80	92	70	68
44	27	47	38	81	100	92	66
55	42	40	50	98	94	65	100
51	33	22	46	98	98	76	98
50	28	67	44	100	92	76	96
49	40	51	42	86	100	60	66
35	28	44	23	93	100	73	68
46	23	57	28	80	94	64	74
48	20	33	32	86	92	64	72
48	24	44	42	98	98	76	79
38		38		100		70	

TABLE II.
THEORETICAL *t* VALUES

Degrees of freedom	Probability			
	0,1	0,05	0,01	0,001
∞	1,65	1,96	2,58	3,29

ACKNOWLEDGMENT

The authors thank anonymous FedCSIS 2014 reviewers for their comments and suggestions. In addition, the authors thank the professor emeritus Leo Võhandu, the Senior Research Scientist of Tallinn University of Technology, for his support and contribution to the present research.

REFERENCES

[1] ECDL Foundation. Retrieved from: http://www.ecdl.com
[2] Felder, R.M. and Brent, R. Understanding Students Differences, Journal of Engineering Education, 2005, 94(1), pp. 57-72. http://dx.doi.org/10.1002/j.2168-9830.2005.tb00829.x
[3] Felder, R.M. and Silverman L.K. Learning and Teaching Styles in Engineering Education, Engr. Education, 1988, 78(7).
[4] Felder, R.M and Soloman, B.A (n. d.) Learning styles and strategies. Retrieved from: http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSdir/styles.htm
[5] Felder, R.M. and Spurlin, J. Applications, Reliability, and Validity of the Index of Learning Styles. Intl. Journal of Engineering Education, 2005, 21(1), pp. 103-112. Retrieved from: http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSdir/ILS_Validation%28IJEE%29.pdf
[6] Khan Academy. Retrieved from: http://www.khanacademy.org
[7] Kolb David. "Experiental Learning", Engle Cliffs, Prentice Hall, 1984, 256 p. Retrieved from: http://academic.regis.edu/ed205/kolb.pdf
[8] Mironova, O., Amitan, I., and Vilipõld, J. Computational Thinking and Flexible Learning: Experience of Tallinn University of Technology. Lecture Notes in Information Technology; 2012, 23-24, pp. 183 – 188. Retrieved from: http://www.ier-institute.org/2070-1918/lnit23/v23/183.pdf
[9] Moodle. Retrieved from: https://moodle.org
[10] Soloman, B. A. and Felder, R.M. (n. d.). Index of Learning Styles Questionnaire. Retrieved from: http://www.engr.ncsu.edu/learningstyles/ilsweb.html
[11] Studies Information System, Tallinn University of Technology. Retrieved from: http://ois.ttu.ee

Appendix C

Mironova, O.; Amitan, I.; Vendelin, J.; Vilipõld, J.; Saar, M. (2016). Maximizing and personalizing e-learning support for students with different backgrounds and preferences. Interactive Technology and Smart Education, 13 (1), 19−35, 10.1108/ITSE-09-2015-0025.

Maximizing and personalizing e-learning support for students with different backgrounds and preferences

Received 3 September 2015
Revised 14 September 2015
17 September 2015
Accepted 30 September 2015

Olga Mironova
Tallinn University of Technology, Tallinn, Estonia, and
Irina Amitan, Jelena Vendelin, Jüri Vilipõld and Merike Saar
Department of Informatics, Tallinna Tehnikaulikool, Tallinn, Estonia

Abstract

Purpose – This paper aims to present a teaching approach to achieve the most personal support for students with different backgrounds and preferences in studying an Informatics course.

Design/methodology/approach – The presented methodology is based on the main principles of flexible and blended learning. The authors considered three main aspects: student's background, the level of knowledge and the most suitable style of learning. At the beginning of the course, students were randomly divided into reference and test group. The test group students were continuously supported by different tools within an e-learning environment. The learning process for the reference group students was held in a traditional form.

Findings – The success of the experiment presented in this paper is demonstrated by comparing the results of the test group who were taught using the new strategy with the reference group who were taught in a common way. The statistical analysis shows that the test group students had better achievements compared to the reference group.

Research Limitations/implications – This presented study was carried out with non-IT first-year university students from social sciences, economics and technical faculties. Each year the number of students varied from 150 to 300.

Originality/value – Based on developed methodology, the model of individualization of the educational process in an e-environment was created and implemented in the course of Informatics in Tallinn University of Technology.

Keywords Curricula, Teaching methods, E-Learning

Paper type Research paper

1. Introduction

High-level higher education is one of the most important aspects in the social development of a country. How can a learner get it? What can an educator do to provide it? First of all, both parties should be aware and using modern techniques that help raise learning effectiveness.

At present, there are so many opportunities to deliver information and knowledge, especially with the help of the World Wide Web. The Internet has quickly spread into all areas of human life around the world. In Estonia, it is impossible to imagine an everyday activity without some e-components: e-government, e-school, e-health, etc. Furthermore, e-learning is also a rapidly developing possibility with various platforms.

Interactive Technology and Smart Education
Vol. 13 No. 1, 2016
pp. 19-35

1741-5659
DOI 10.1108/ITSE-09-2015-0025

The main goal of e-education is to provide knowledge for each learner in the most suitable form. However, huge amounts of information from different sources do not guarantee perfect knowledge. Therefore, educational e-materials have to be carefully selected and structured to provide for the needs of the learners. It is a new paradigm of the present-day education.

In modern times, all information and, as a result, all knowledge is rapidly changing, making it impossible for educators to predict exactly what the contemporary student will need to know tomorrow – on the next level of his study or in their future work. Instead, teachers have to be helping students to develop the necessary habits and skills which would enable students to learn whatever they might need in their future.

At the moment, Informatics is a continuously and rapidly changing subject in the university curriculum. The main purpose of computing courses, therefore, should be to develop computational thinking and skills needed to use the standard personal computer (PC) equipment on an advanced level, as well as to help users understand and follow the latest trends in the field of information technology (IT).

The contents of an Informatics course often seem to be inextricable for the first-year non-IT students. There are several reasons for it: from low motivation to the absence of the real IT projects during the first year at the university. Gradually, it has become obvious that IT courses need to be innovated and the needs of the target audience have to be taken into account: we need more flexible learning possibilities and as much individualization in the teaching process as possible.

2. The course outline

The Informatics course under discussion belongs to the curriculum of all specialities and it has been designed for the first-year non-IT students. It lasts for two semesters and is taught in two languages: Estonian and English.

The main purpose of this course is to develop computational thinking by means of creating applications (using spreadsheets and some programming environments). The learning process starts with processing and analysing data in MS Excel: writing formulas, using built-in functions and drawing charts. The amount and topics of practical tasks and the theoretical material depend on the students' faculty: social sciences, economics, chemistry, civil engineering or mechanics.

In the second part of the course, students learn the basics of modelling and programming in practice. Python and Visual Basic for Application (VBA) are the programming languages used.

It should be mentioned that this part of the Informatics course used to be rather complicated for most of the students, especially students from the economics and social sciences departments. Difficulties at the start of programming tasks were solved by implementing graphical programming as an introduction to the course. After several years of experiments, we chose the graphical programming environment Scratch as the main tool to introduce the main programming steps (MIT Media Lab, 2015). Scratch is a very intuitive language and greatly helps learners to take on board the main concepts and terms of modelling and programming such as the variable, subroutine, process, branching or cycle.

During the course, we try to use and combine different styles of teaching and learning: the orthodox face-to-face classroom method, pair or group work and, especially, students' independent learning in the e-environment. A learning environment such as Moodle

(Moodle Trust, 2015) provides us with a great variety of different additional opportunities, which enable making the learning process more flexible and more individual. A major way is the learning pace adjustment and variety in the number of learning assignments, which are based and matched with students' level of knowledge and their learning styles.

The main learning outcomes of the Informatics course are listed below. Students who complete the course:

- acquires the foundations of independent working skills;
- can justify the feasibility of tools and methods chosen for problem-solving;
- is familiar with the principles of creating applications as well as the methods and tools used for it, and with the principal phases of the development process;
- is familiar with the capabilities of mainstream application software and environments and can use these (spreadsheet software) efficiently to create applications;
- Can use different types of data, also formulas, expressions and internal functions of spreadsheet software (Excel) to solve problems arising in the field of engineering;
- can create applications by using spreadsheet programs consisting of multiple linked tables that use formulas, validation and lookup functions as well as table processing and data analysis tools;
- is familiar with the principles of programmatic control, has a general idea of the nature of processes occurring in programs and can create simpler programs in VBA;
- acquires the foundations of problem analysis and system modelling;
- can analyse relations between objects and provide 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 can describe using VBA/Python and UML activity diagrams main activities occurring in programs and algorithms;
- is familiar with the nature and main concepts of object-oriented programming;
- can compose programs consisting of multiple procedures and organize data flow between them; and
- can use graphical objects in programs; develop scaled drawings and schemes, movements and animation in VBA.

To reach these goals, the authors try to use modern technics and work out a teaching model that could help us to provide each learner an opportunity to participate in the learning process maximally effectively.

3. The course evolution

3.1 The background

Unfortunately, teachers cannot provide all learners with one-to-one tutoring at all educational institutions. However, this fact has not affected the main aim of the educational process – to guarantee high-quality knowledge and modern skills. During

the experiments with our course structure and content, we considered the differences in students' characteristics, especially their background (the faculty and the level of prior knowledge) and their preferred learning styles.

Since 2010, a group of lecturers from Tallinn University of Technology started applying a new approach to the design of the computer science courses. Year by year, this approach has become more flexible and adaptive to the nature of every student.

At the beginning, we randomly divide all the students into equal reference and test groups. The division is not linked to the students' specialization. The average number of members in each group is about 100-150; it varies depending on the general number of students at the university. It should be noted that students are not aware of the research. The reference group is taught using the same course materials but these students are not supported with any additional systems. The students of the test group are directed in choosing their e-learning materials based on the data obtained through the tests in the e-environment.

Our intention here is to compare the results of these two groups at the beginning and at the end of the course.

At the beginning of the course, the students are tested to find out their level of knowledge in the Informatics subject. Experience has shown that such testing is necessary for the development of the course content. The purpose is to keep track with new times and main trends, as the computer science is one of the fastest-developing sciences (College Board, 2015; Computer Science Teachers Association, 2015). At the same time, it is necessary to follow the situation in our country schools to have an opportunity to predict the level of knowledge of future students.

The nature of the tests for both groups is similar and based on the concepts defined in the European Computer Driving License (ECDL Foundation, 2015). The assignments focus on some principles of the work with the PC like creating text documents and presentations, handling information using the spreadsheets and elementary programming knowledge. Tests include both practical and theoretical tasks.

The programming category of the questions was added to the test some years ago and is currently developing rapidly. We bear in mind that a new elective course "Basics of Application Development and Programming" was recently included in the secondary school curriculum in our country and assume that this is already showing the first results. This course was designed and worked out at the Department of Informatics of Tallinn University of Technology to raise the computing skills among the school pupils in Estonia.

The current situation of teaching computer sciences at Estonian schools is quite discrepant. Some secondary schools do not have Informatics lessons at all, and in some schools, it is taught only for two or three years, which is a very short period to prepare students for the university level. This drawback is associated with two main reasons. The first one being that there is no nationwide Informatics curriculum in our country. The second one is that Informatics subjects are not mandatory in our primary and secondary schools. A logical consequence of these reasons is the situation where each school teacher introduces learners to the material at his own discretion and style: certain pupils draw in Paint, others learn the computer hardware in theory, etc.

In connection with this, the level of PC skills among non-IT learners falls every year and reduces to commonplace Facebook usage. The follow diagram (Figure 1) shows the last year statistics about non-IT testing at the beginning of the Informatics course. The

feasible maximum number of points is 100. As can be seen, the level of computer skills is quite low and steadily decreasing.

We are also faced with another problem: school pupils who have obtained sufficient Informatics knowledge at their schools or additional courses often become IT students at university level. Unfortunately, they are not our audience.

In the current work for innovating the Informatics course and individualization of the learning process, the authors can name the main strategy – a maximally effective use of the e-environment with its possibilities. E-learning allows modern students (with a personal computer, connected to the internet) to attend the course anywhere and at any time. It is very important to get the material or submit the homework on time regardless of whether the student is or is not at the university.

3.2 E-course

The first phase, which we named "e-course", includes the adaptation of educational materials for the e-environment in Moodle.

Uploading a set of lecture materials and exercises into a learning environment does not ensure that students comprehend it and obtain necessary knowledge. Therefore, to make study materials suitable for an e-course, all teaching materials (theory as well as materials for practice) were thoroughly revised. Our aim was to provide an effective delivery of the online content. To achieve this goal, we aimed at working out a new pedagogical and didactic policy as well as strategies for the new e-course.

Theoretical materials were innovated and supported with learning videos. It should be noted that in our course, we use not only video lectures but also short screen-captures, which explain the most complicated tasks. Creating these videos, the authors adhered to the principles of Khan Academy (2015). Practical tasks of the course were reconsidered and supplemented with various group and pair-work tasks and also self-tests. These innovations made the Informatics course more dynamic and attractive for our students. Both groups, the test and the reference group, got access to this renewed course.

At the same time, we have not abandoned the standard lessons in computer classes. They were held as usual but now we got many advantages. Because of the e-lectures and visual explanations in Moodle, we had more time for practical training in contact lessons. It is necessary to mention that students have no access to practical exercises unless they solve the tests which are based on the theoretical material of each topic. Thereby students come to the lessons already prepared for the practical tasks. Often

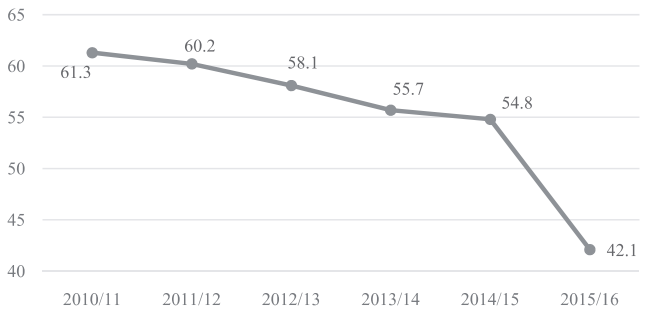

Figure 1.
The averages of the beginning test

they get a few small practical tasks in the e-environment and afterwards, in class, they use already ready solutions to solve the bigger tasks.

It is generally known that effective computing is impossible without practice. During face-to-face lessons, students work in pairs or groups and are given an opportunity to try the obtained knowledge in practice. Moreover, such kind of work develops teamwork skills, which is very important, especially for the first-year students during their first semester.

In such case, the role of the classical educator is slightly different – the lecturer becomes more of a supporter in the students' team work of their learning assignments. During contact lessons, learners have an opportunity to ask questions related to their homework and share their practical skills and experience with the rest of the group. Practical knowledge transferred in such a way is obtained much faster than in standard lectures. This fact was confirmed by the results of student tests and feedback.

In their feedback, students named another advantage of such practical lessons: they have an opportunity to get support or to ask something not only from their teacher but from other students too. It should be mentioned that this form of support is equally important and useful for both sides: the one who gets it and, especially, for the one who gives it. To detect, explain and, afterwards, to correct a mistake in calculations or in the program code is a substantial practical skill in computer science subjects.

In addition, in an e-environment, students get their practical assignments in accordance with their specialities but are still united under a common subject topic. For example, students from the economics department get more tasks related to table calculations; social sciences students implement the information filtration, statistics calculations and various requests.

It should be noted that the course materials are organized sequentially and it is not possible to get a new portion of theoretical materials and practical tasks without solving the previous ones. Thanks to checking opportunities in the e-environment, like tests or self-tests, teachers do not have to spend time on routine inspection of the assignments at all. Using the automated checking tools gives students an opportunity to learn within their own pace. They do not have to wait for the feedback on the assignments from the lecturer and his/her manual permission to proceed onto the next level. E-checking systems do it faster and as many times as is needed.

In addition to the aspects mentioned above, we and our learners use forums very actively in our course. This way the students get an opportunity for online communication and fast online help. Our aim was to show that they can get support and advice any time, and can also share their ideas, problems and solutions. Through these forums, we often get perfect brainchildren for group work and individual assignments.

Another advantage that non-IT students have when they participate in the e-course is that their technical skills improve.

Transition to the Moodle e-environment gave our educators an opportunity to follow students' progress and it became fairly easy to get the statistical data of different samples for analysis, development and improvement of the implemented learning methods.

It makes no sense now to recount all the advantages that we achieved during the first stage of our modification of the course. Such benefits have already been systematized and described in detail in different sources (Broadbent, 2002). However, we would like to mention the first positive results of our work: students' feedback and increase in

academic achievements. The data were extracted from an e-environment, where students and teachers get information about the courses and curricula, students declare courses, keep results and give anonymous feedback on their educational process. The authors do not present specific data in numbers in this paper because the transition of the course into e-learning took place a long time ago.

3.3 Prior knowledge

Starting from the second stage, which the authors entitle "prior knowledge", experimental work takes place only with students from the test group. The students of the reference group are taught as usual.

At the beginning of the course, we start with dividing the students into three e-streams based on their readiness for Informatics subjects. This division was realized through an e-test and implemented in the Moodle environment. The students, however, were not aware of the experiment.

In the described division, we proceeded from the level of students' knowledge required to start the course. Those students whose e-test results are more than 85 points we named "experts"; "advanced users'" result is between 60 and 80 points, other students are called "beginners".

The named groups of students receive different amounts of practical and theoretical tasks in Moodle, with different levels of difficulty. To move to the next topic, the mandatory set of exercises has to be solved. In the e-environment, the "beginners" and "advanced users'" have to solve their sets of tasks before they get access to "experts'" exercises – the main material of the course curriculum. "Advanced users" solve their tasks and can proceed to the main topics (Figure 2).

These additional sets of tasks are catered for exactly what learners need to know for the current Informatics course. Students do not need to pay for any additional IT courses and they get all the materials and assignments centrally, in one place – the Moodle e-course, in parallel with their main studies.

To automate and speed up the checking of the increased number of tasks, we have developed a special e-tests system in the Moodle.

Thereby, we could increase the amount of practical assignments for students with different levels of readiness without increasing the subject hours and students' load. This stage of the course innovation gave us an appropriate level of the students' readiness for face-to-face lessons.

The above-mentioned method provided our Informatics teachers with actual and important information about what the learners knew before starting the course. Every

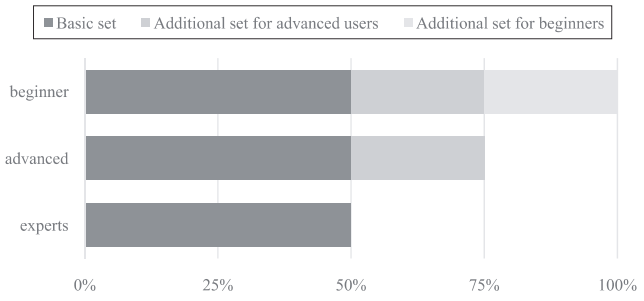

Figure 2.
The distribution of practical assignments

year we get an overview of the current situation of Informatics subjects at secondary schools in our country. Moreover, according to the results, we are able to provide students with the necessary learning materials.

3.4 Learning style

The third part of our innovation experiment with the course modification is the "learning style" phase and it was also realized in Moodle. This stage is a continuation of the test group's students division into groups. There is a lot of research about the individualization of learning depending on students' abilities (Kolb, 1984; Palmer, 2011; George Lucas Educational Foundation, 2014). In our course, Felder-Silverman model was picked out as the basis for the distribution (Felder and Spurlin, 2005; Felder and Brent, 2005). In this stage, we could maximize the use of the Moodle e-opportunities.

Learning styles are characteristic cognitive, affective and psychological behaviours that serve as relatively stable indicators of how learners perceive, interact with and respond to the learning environment. Students learn best when instruction and learning context match their learning style.

Depending on their learning style, Felder differentiates between the following groups of learners:

- active and reflective;
- sensing and intuitive;
- visual and verbal; and
- sequential and global.

For better comprehension of the topic, we give a brief description of the learners' types.

Active learners acquire new knowledge best by doing, discussing and explaining it to others in a group. At the same time, reflective learners first think about it alone.

Sensing learners like learning facts and solving problems by well-known methods. Intuitive learners prefer discovering new possibilities and relationships and they are more innovative.

Visual learners remember pictures, diagrams, charts and videos best. Verbal learners prefer written and spoken explanations.

Sequential learners like step-by-step studying, where each step follows logically from the previous one. Global learners prefer to get information by large portions and randomly.

Through a test, we learned that the majority of our course participants in the test group were active and visual learners and they had very strong preferences for their learning process. These preferences were detected according to the Felder test, which was held at the beginning of the course. It should be noted that each year, the number of active and, especially, visual students increases (Figure 3).

Throughout the educational process, students were provided with necessary learning materials and activities in accordance with Felder's instructions.

For example, active learners automatically received more group work and opportunities to help others – they could check and correct other students' work and assignments in Moodle; they answered questions in the e-course forums and took the role of a tutor in face-to-face classes. It should be mentioned that they did it with pleasure. For visual learners, a great variety of visual representation of the educational

materials (that was already mentioned above) was provided. Interests and preferences of the other types of learners were also taken into account.

4. Programming basis for visual learners

As mentioned above, the majority of the course participants are visual and active learners. For visual learners, it is very important to see "how it works" and for active – "to try it out". In the most complicated module of the Informatics course, programming, we found a solution – maximum visualization.

Our introductory module, with Scratch as the main tool with its graphical elements, is already perfectly visualized. Scratch's dynamics and attractiveness helps visual students to understand the main ideas of the created applications. Working in this programming environment, active learners can manually modify some program elements and immediately see the result. Moreover, there are no syntax mistakes in Scratch that greatly facilitates the programming process. Therefore, introduction to programming in our course has already been visualized (Figure 4).

The next module, VBA or Python, is rather complicated for non-IT students. What can be visualized there?

VBA already has a built-in visualizing tool: students can follow the code execution using the Locals Window (Figure 5). The named window automatically displays all the declared variables in the current procedure, their names, types and values in the real time. When the Locals Window is active and visible, it automatically updates during the time the program is running – visual students can follow the code and check each step and its result in their applications, especially if they run programs in step-by-step mode.

Besides the Locals Window, it is possible to use the Immediate Window to reach the same goal: check and correct mistakes and understand the program structure.

Unfortunately, Python does not have such an opportunity, but in spite of this, it still needs to be visualized for the beginners in programming. Therefore, we introduce the Online Python Tutor to our students. Using this internet application, they can follow each step of their program code and check the variables' values, types and the order of the operators during the execution. It has to be mentioned, though, that the online tool has some drawbacks, as it does not support the Python graphics, time functions and files processing. However, for the beginners in coding the application gives an understanding of the main principles of program construction and code execution (Figure 6). There have been cases when students started to write the program not in Python environment but directly there, on the internet.

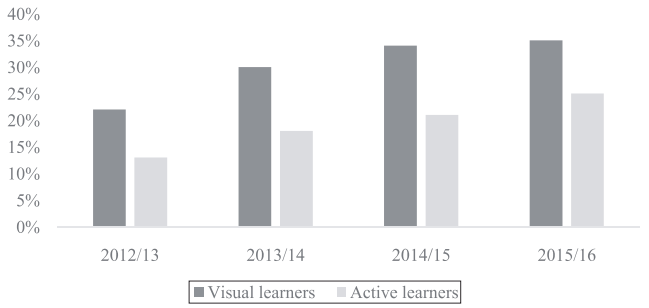

Figure 3.
The increase of the number of the visual and active learners

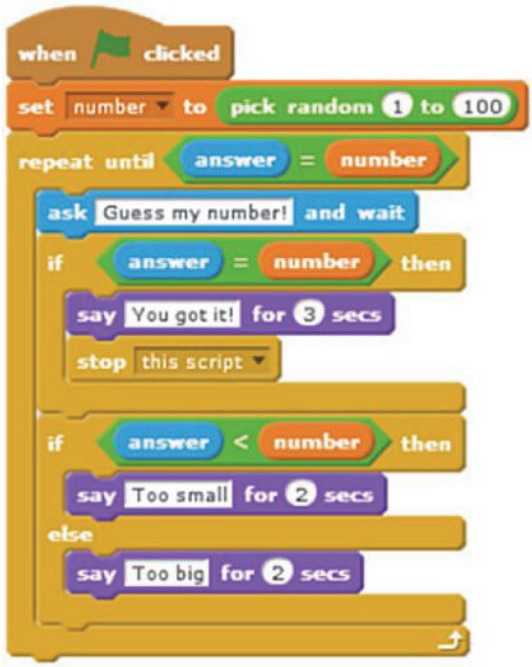

Figure 4.
Scratch program

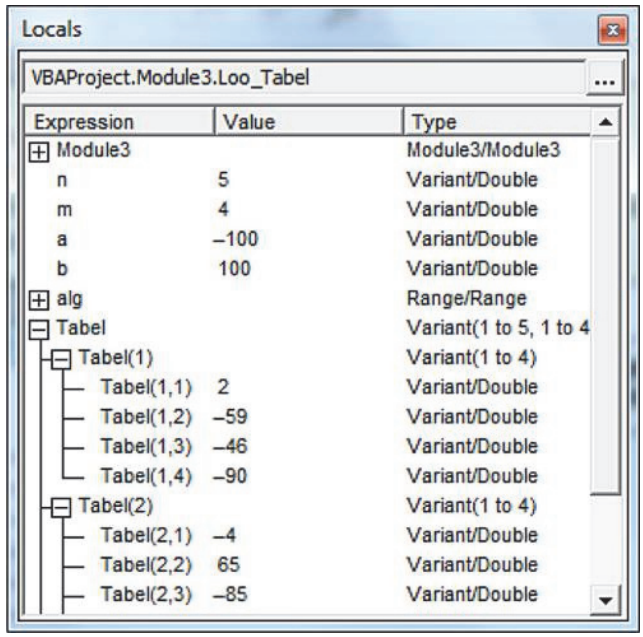

Figure 5.
VBA visualizing tool – Locals Window

Figure 6.
Python visualizing tool

Using these online and built-in tools, we could raise the interest to programming among the non-IT students in the second part of the Informatics course.

5. Comparing the results

5.1 The method of analysis

Throughout the described experiment, good exam results and positive students feedback showed us the positive effect of the curriculum and course modifications. Then it was decided to examine the data with statistical methods. The aim of this examination is to check the correctness of the chosen approach and direction to an educational process.

As the method of the hypothesis testing, the Student's *t*-test for the comparison of the two means was chosen. This test assumes a normal distribution of samples and not significant differences between the standard deviations of either sample. Our aim is to show that there were no significant differences between the test and reference group students in September 2014, while in January 2015, the results are significantly different.

For calculations, we use the equations for the averages $\bar{x}$ and corresponding standard deviation S for both groups (2):

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{1}$$

$$S = \sqrt{\frac{\sum_{i=1}^{n} (\bar{x} - x_i)^2}{n - 1}} \tag{2}$$

After that, we calculated the standard error σ using the equation (3):

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (\bar{x}_{Test} - x_{iTest})^2 + \sum_{i=1}^{n} (\bar{x}_{Ref} - x_{iRef})^2}{n \cdot (n - 1)}} \tag{3}$$

Finally, using equation (4) we calculated the experimental value $t_{\exp}$:

$$t_{\exp} = \frac{|\bar{x}_{Test} - \bar{x}_{Ref}|}{\sigma} \quad (4)$$

In addition, to compare this value with theory, we need to calculate the degree of freedom *df* using equation (5):

$$df = 2n - 2 \quad (5)$$

As initial data for calculations, we chose September 2014 students' test results of the test and reference groups and the same groups' results in January 2015 (Table AI in the Appendix).

Both samples are in the equal size: $n = 75$ and distributed normally (Figures 7 and 8).

The two means and the corresponding standard deviations are calculated by using the equations (1) and (2):

$$\bar{x}_{Test} = \frac{2898}{75} \approx 38{,}64$$

$$\bar{x}_{Reference} = \frac{2912}{75} \approx 38{,}83$$

$$S_{Test} = \sqrt{\frac{22793{,}28}{75 - 1}} \approx 17{,}55$$

$$S_{Reference} = \sqrt{\frac{23010{,}75}{75 - 1}} \approx 17{,}63$$

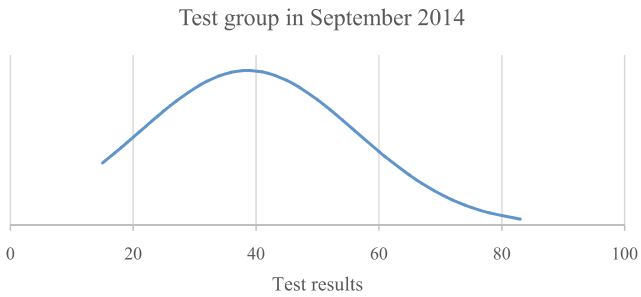

Figure 7.
The distribution of the test group results

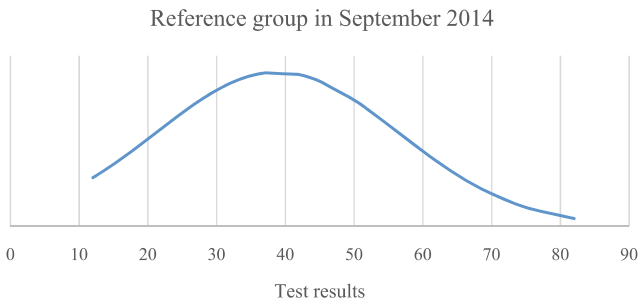

Figure 8.
The distribution of the reference group results

Now we see that there is no significant difference between the standard deviations in either groups. It means that we can continue with Student tests.

The standard error of the difference between the two means is calculated by using equation (3):

$$\sigma = \sqrt{\frac{22793{,}28 + 23010{,}75}{75 \cdot (75 - 1)}} \approx 2{,}87$$

Experimental t -value is calculated using equation (4):

$$t_{\exp Sept} = \frac{|136{,}64 - 38{,}831|}{2{,}87} \approx 0{,}06$$

To compare this value with the theoretical t_{th}, we need to calculate the degree of freedom using equation (5):

$$df = 2 \cdot 75 - 2 = 148$$

Using equations (1) to (4), we then calculated both groups' results in January 2015:

$$\bar{x}_{Test} = \frac{6687}{75} \approx 89{,}16$$

$$\bar{x}_{Reference} = \frac{5621}{75} \approx 74{,}95$$

$$S_{Test} = \sqrt{\frac{6622{,}08}{75 - 1}} \approx 9{,}46$$

$$S_{Reference} = \sqrt{\frac{12161{,}79}{75 - 1}} \approx 12{,}82$$

$$\sigma = \sqrt{\frac{6622{,}08 + 12161{,}79}{75 \cdot (75 - 1)}} \approx 1{,}84$$

$$t_{\exp Jan} = \frac{|89{,}16 - 74{,}95|}{1{,}84} \approx 7{,}73$$

5.2 Results of the experiment

Using the table of theoretical t_{th} values with the corresponding degree of freedom (Table AII in the Appendix), we found that the means of September results are not different at any critical level:

$$t_{\exp Sept} < t_{th}$$
$$0{,}06 < 1{,}65$$

This means that at the beginning of the course, both groups of students, the test and reference, had the same level of knowledge.

January 2015 results are the opposite – the means are different at critical levels:

$$t_{\exp Jan} > t_{th}$$
$$7{,}73 > 3{,}29$$

It shows that the students who were taught using our system of the learning process individualization obtained knowledge much better than the others.

The progress of both groups is shown in the Figures 9 and 10.

These results confirm the validity of our study and the chosen method of course individualization.

6. Discussion

This research investigates how the considered three-stage individualization of the learning process positively affects the students' knowledge, motivation and academic progress, as a result.

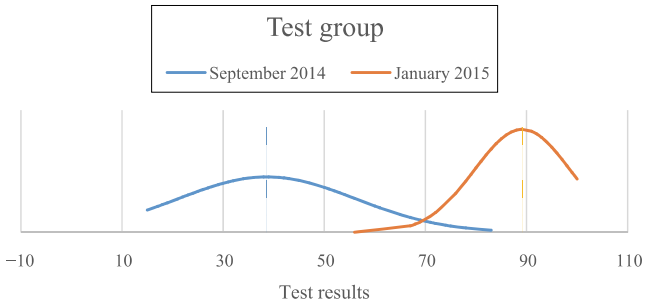

Figure 9.
The test group progress

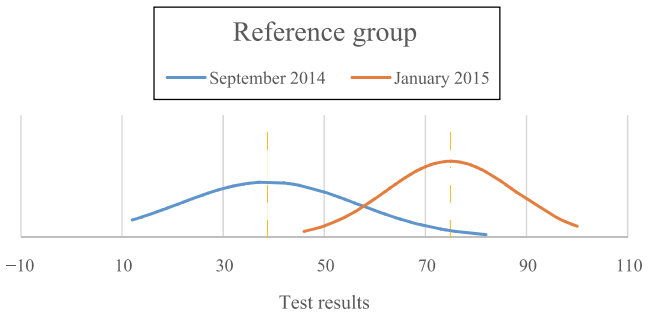

Figure 10.
The reference group progress

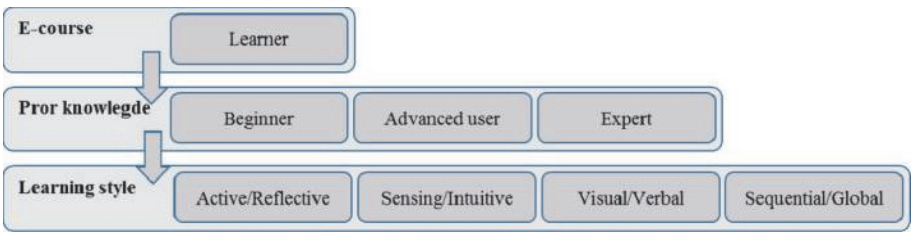

Figure 11.
The model of the course

The two main findings are presented in this paper: prior knowledge and learning style and their integration and collaboration. After course transition to the e-learning environment, it is firstly important to take into account students' prior knowledge to provide them the required additional support. Second, to consider how do they learn better to provide the understanding and knowledge obtaining in the most effective way.

Classroom activities of teachers and students took place in mutual communication. Therefore, the guidance and the formative role of the teacher were realized in the creation and review of the theoretical material and the material in practical classes.

On the other hand, the supportive role of the instructors of the presented Informatics e-course becomes less noticeable and unostentatious for students. It gives learners the opportunity to feel some kind of independence in learning process thereby greatly increasing their motivation and learning activity.

During the process of upgrading the Informatics course, authors were able to create a model of individualization of the educational process in an e-environment, which considers the level of students' prior knowledge and their preferences in the learning process (Figure 11).

Applying this model, instructors try to find an individual approach to each student in the Informatics e-course and make it more flexible. Furthermore, as shows the anonymous feedback, it was possible to make the Informatics course more interactive and more attractive for the first-year students.

The main results of presented research are valid and reliable, as they have been statistically examined and not based only on observation.

7. Conclusions and future work

The model of blended and flexible learning, presented in the article, shows positive outcomes and trends of the strategy and suggests that there are no unreachable aims in an educational process. The comparison of two relevant groups, the test and reference, demonstrates significant differences in achievements at the end of the first part of the course. The feedback, received from the groups, was also different: the reference group was more motivated and interested in the learning process.

The authors intend to continue with the created model and develop the Informatics e-course for non-IT students in the chosen direction, trying to maximally adapt it to students with different preferences and backgrounds.

The next step of the present research is the extension of the model of individualization to the first-year IT students' courses aimed at continuous improving the quality of education in Tallinn University of Technology.

References

Broadbent, B. (2002), *ABCs of e-Learning: Reaping the Benefits and Avoiding the Pitfalls*, 1st ed., John Wiley & Sons, New York, NY.

College Board (2015), "AP computer science principles", available at: https://advancesinap.collegeboard.org/stem/computer-science-principles (accessed 4 May 2015).

Computer Science Teachers Association (2015), "Computational thinking task force", available at: http://csta.acm.org/Curriculum/sub/CompThinking.html (accessed 2 May 2015).

ECDL Foundation (2015), "ECDL foundation", available at: www.ecdl.com (accessed 1 May 2015).

Felder, R. and Brent, R. (2005), "Understanding students differences", *Journal of Engineering Education*, Vol. 94 No. 1, pp. 57-72.

Felder, R.M. and Spurlin, J.E. (2005), "Applications, reliability, and validity of the index of learning styles", *International Journal of Engineering Education*, Vol. 21 No. 1, pp. 103-112.

George Lucas Educational Foundation (2014), "Project-based learning", available at: www.edutopia.org/project-based-learning (accessed 24 November 2014).

Khan Academy (2015), "Khan academy", available at: www.khanacademy.org/ (accessed 3 May 2015).

Kolb, D. (1984), *Experimental Learning*, Prentice Hall, Engle Cliffs.

MIT Media Lab (2015), "Scratch – imagine, program, share", available at: http://scratch.mit.edu/ (accessed 6 May 2015).

Moodle Trust (2015), "Open-source learning platform | Moodle.org", available at: https://moodle.org/ (accessed 6 May 2015).

Palmer, S.R. (2011), "The lived experience of flexible education – theory, policy and practice", *Journal of University Teaching & Learning Practice*, Vol. 8 No. 3.

Corresponding author
Olga Mironova can be contacted at: olga.mironova@ttu.ee

Appendix

September 2014 Groups' results				January 2015 Groups' results			
Test		Reference		Test		Reference	
45	17	34	19	72	67	75	75
67	53	54	43	88	96	81	76
53	23	26	23	79	88	75	76
67	24	21	28	95	94	79	87
19	29	17	29	89	56	75	98
17	29	43	65	98	89	80	99
31	27	29	27	89	99	80	57
43	34	31	53	88	100	62	68
39	31	29	32	85	91	75	87
54	31	28	31	85	87	65	67
72	51	37	42	100	97	86	77
83	71	19	45	89	98	60	78
78	43	82	34	87	98	100	87
21	22	45	12	86	98	75	87
22	22	52	22	98	69	76	76
22	31	61	45	100	79	100	75
32	38	32	19	99	79	64	65
22	38	32	19	93	79	56	63
24	52	43	34	96	68	50	79
25	19	54	24	97	98	98	86
51	21	21	21	95	94	66	87
42	27	34	28	96	95	75	87
43	34	42	32	78	97	85	76
43	65	19	64	98	97	65	56
43	61	19	45	100	96	67	65
17	50	67	50	86	94	69	68
32	60	32	60	80	91	69	87
43	34	54	32	81	93	71	76
25	69	54	61	82	95	54	76
27	47	37	35	85	94	67	56
27	45	37	35	84	100	71	71
74	39	44	62	76	100	72	46
36	75	56	18	78	76	74	63
56	15	76	17	79	79	81	70
41	26	76	69	94	83	82	56
21	25	14	73	93	89	100	100
21	24	21	74	92	98	54	88
45		34		72		75	

Table AI. The test results of both groups

Degrees of freedom	Probability			
	0.1	0.05	0.01	0.001
∞	1.65	1.96	2.58	3.29

Table AII. Theoretical *t*-values

Appendix D

Mironova, O.; Amitan, I.; Vendelin, J.; Vilipõld, J.; Saar, M. (2015). Object-Oriented Programming for non-IT Students: Starting from Scratch. International Journal of Engineering Pedagogy, 5 (4), 22−28, 10.3991/ijep.v5i4.4734.

Object-Oriented Programming for non-IT Students: Starting from Scratch

http://dx.doi.org/10.3991/ijep.v5i4.4734

O. Mironova, I. Amitan, J. Vendelin, J. Vilipõld and M. Saar
Tallinn University of Technology, Tallinn, Estonia

***Abstract*—The present paper demonstrates a teaching approach in general programming course for the first year non-IT students at Tallinn University of Technology, Estonia. The authors suggest some ways for achieving better results in programming issues that are usually complicated for the beginners.**

***Index Terms*—object-oriented programming, Scratch, VBA, Python.**

I. INTRODUCTION

Information technology and programmable systems, such as computers, smartphones and other devices, are playing an increasingly greater role in our lives today, both at home and at work. We can see growing demand for information technology professionals and escalating need for the knowledge about computer technology for other specialists. "Computational thinking" is a skill that all students must learn to be ready for the workplace and able to participate effectively in the digital world.

Basic computer education topics have been included into all the curricula at Tallinn University of Technology, Estonia, and have been integrated into a course named "Informatics".

The Informatics lasts two semesters and number of weekly study hours is two. Group size is 20-30 students. During the course we apply classic face-to-face classroom methods, group work and learning in the Moodle e-environment.

The main learning outcomes in the Informatics course are listed below. Students who complete the course:

- Acquires the foundations of problem analysis and system modelling.
- Can analyse relations between objects and provide 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 can describe using VBA/Python and UML activity diagrams main activities occurring in programs and algorithms.
- Is familiar with the nature and main concepts of object-oriented programming.
- Can compose programs consisting of multiple procedures and organize data flow between them.
- Can use graphical objects in programs; develop scaled drawings and schemes, movements, and animation in VBA.

The course starts with application development in the environment of general-purpose application software such as document and spreadsheet processing. The second part is devoted to building algorithms and programming. The aim of this part is to develop logical, analytical and algorithmic reasoning skills as well as the ability to investigate problems and tasks in a systematic way.

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

It should be mentioned that the Informatics course seems to be rather sophisticated for most of the non-IT students. The main issues in teaching the subject have been delineated and systemized in [1], [2]. However, we still face some problems. Consequently, we try to improve the course content from year to year and from speciality to speciality to find the best methods to achieve the goals.

After several years' experience two main algorithmic languages were chosen for creating applications. These are Python and Visual Basic for Applications (VBA). We have different reasons why we prefer one or the other, but there are a lot of things that should be considered and mastered beforehand.

II. THE REVIEW OF THE CURRENT SITUATION IN THE COMPUTING TEACHING

In recent years, several countries have carried out thorough investigations of the use of information technology and courses on computer science in different schools. Analyses have shown that most of the courses do not meet the needs. As a result, several new curricula have been proposed to improve the situation.

In 2011 the new CS standard, "CSTA K-12 Computer Science Standards" was created [3]. It sets out the basic requirements for the various areas and levels of the curricula. A number of courses and subject syllabuses were created on this basis. One of the most outstanding is the new CS syllabus "AP Computer Science Principles" [4] created under the support of US National Science Foundation (NSF). The work started in 2011 and the course is planned to be completed in 2016.

The documents mentioned above are based on the notion of "Computational Thinking", which defines general principles for describing the problems and solving them by means of software systems, including such concepts as abstraction and modelling, algorithms, data and information, programming, communicating and collaborating. A large part of the concepts is related to algorithms and programming [5].

In 2012 a comprehensive study "Shut down or restart?" was published by The Royal Society UK [6]. The research brought out significant shortcomings and offered ways to solve them. "Computer Science: A Curriculum for Schools" [7] was set up and published, where "Computational Thinking" is the main idea. Starting from September 2014 the course Computing (Computer Science + Information Technology + Digital Literacy) is included in the UK school curricula.

In our teaching approach to the programming course we use the principles introduced above and try to implement them in the best possible ways.

The current situation of teaching computer sciences at our country schools is quite discrepant. Some schools do not have informatics lessons at all, in some schools it is taught only for two or three years, which is a very short period to prepare students for the university level. This drawback is associated with two main reasons. The first one being that there is no nationwide Informatics curriculum in our country. The second one is that Informatics subjects are not mandatory in our schools. A logical consequence of these reasons is the situation where each school teacher introduces learners to the material at his own discretion: certain pupils draw in Paint, others learn the computer hardware in theory, etc. In connection with this, the level of PC skills among non-IT learners falls every year and reduces to commonplace Facebook usage. In our course we have to take these facts into account and build the curricula accordingly.

III. MODELLING

During the Informatics course students have to create simple applications. After the task is set, we go to the next step: modelling. There are two main aspects to learn: defining the data objects and building algorithms.

The object-oriented approach is the main technique in building and developing software applications and information systems. Its essence is in describing the properties and the behaviour of real and abstract subjects by means of software objects.

The software object is a collection of connected data and programs. The computer system and application software uses object-oriented approach to define documents, user forms, windows, toolbars, etc. The newer programming languages are object-oriented, where the basic concepts are classes, properties, methods and events. The relationships between the objects are used as well.

Aspects containing the description of objects make up a significant part of the application model.

Another part of the modelling process is the description of algorithms used mostly as object methods.

The algorithm is often built step-by-step, starting with general description and becoming more specific later. Finally, we come to a level, where it is easy to transform the outcome into a program code.

UML (Unified Modelling Language) supports building models for applications with a set of diagrams. It helps the learners to reach a solution without reference to any of the programming languages. It is widely used not only in the general programming course, but also later in courses on information systems, databases and several others. However, UML diagrams provide static views and are not always the best to keep track of the process.

IV. VISUAL PROGRAMMING WITH SCRATCH

A new trend in teaching programming skills is the development of an environment created especially for learning. These are graphical tools, such as Scratch [8], Snap! [9], Blockly [10], which make the learning process much easier for the beginners especially for non-IT, who have not any experience in programming.

In our course we use Scratch as the supporting tool before creating applications in VBA or Python. After a few years of practice, we came to the conclusion that a graphical environment, such as Scratch, is an effective introductory tool to understand both the object-oriented approach and the functionality of a program.

In addition, syntax errors are impossible in Scratch, which is a great help for students. It is easy to discover and correct run-time errors as well, because Scratch works as an interpreter.

Graphical command blocks give a visual picture of the different controls (selections, loops), used in the program. They create the necessary associations when students start coding in a text-based programming language.

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

It should be mentioned that according the annual students' feedback Scratch is the most popular module in the course.

Creation of objects is solved by importing or drawing graphics. Combining the blocks for each object creates the methods. 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.

Further, we will briefly review the main programming concepts that are usually complicated for non-IT learners.

A. Objects and their properties

Each Scratch object (Sprite) has properties like the name, coordinates, direction, rotation style, etc. (Fig. 1).

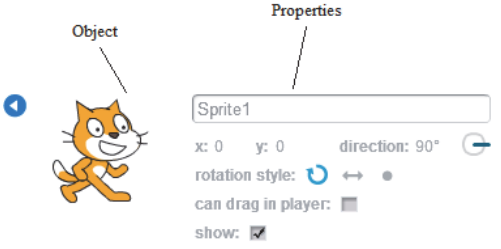

Figure 1. A Scratch object and its properties

Most of such properties can change their values by some method (script) or may be changed manually by a user.

Fig. 2 shows some blocks used to change properties of an object.

Figure 2. Blocks for changing the properties of an object

These blocks make it quite easy to understand the property concept.

B. *Building scripts with conditional statements and iterations*

Using Scratch blocks makes it easy to show students how cycles and if-operators work (Fig. 3).

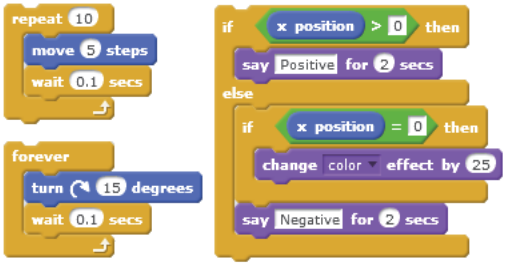

Figure 3. The cycles and branching realization

C. *Events*

There is a rather long list of pre-defined events that can be handled by Scratch scripts. An event is handled by a script (method) starting with a special block. Some of the event blocks are shown in Fig. 4.

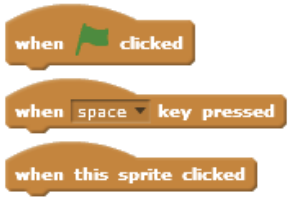

Figure 4. Some blocks to start the event handling script

D. *Parallel processing*

Running each script is considered a separate process. Scripts that handle the same event will be running in parallel after the event occurs. For example, scripts in Fig. 5 are performing in parallel and they implement the reaction on the "click" event. The first of them is responsible for moving the object, the second for changing its appearance.

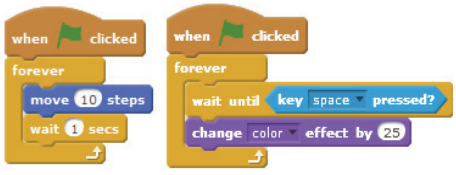

Figure 5. Realization of the parallel processes

Using Scratch scripts, students quickly obtain the concepts of parallel and coherent processing.

E. *Sending and receiving messages*

The message system provides communication and synchronization of the objects' behaviour.

Special command blocks broadcast a message from a script and any other script in the project can pick it up. Each object can send the message to other objects or to itself. The last instance is used to define the structure of an application.

There are two different blocks - one is for broadcasting and continuing, the other for broadcasting and waiting. The latter allows the process to continue only when all processes that picked the message have finished.

Scratch solves reactions to messages in the same way as handling other events. The script reacts to the message if it starts with a special block. (Fig. 6).

Figure 6. Blocks for sending and receiving messages

F. *Data*

Variable is one of the basic concepts in all programming languages. Its meaning in programming differs from its use in mathematics, which learners know from school. Scratch supports clarity in understanding the meaning of a variable concept as a named place in the computer memory. All variables have to be created manually before using them in a program code. Figure 7 shows the command "Make a Variable" in the Data group of the blocks.

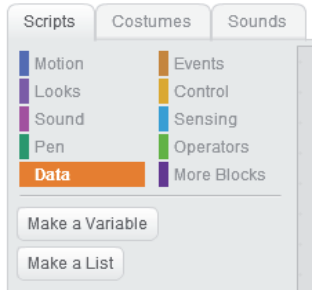

Figure 7. Commands of the Data group

The approach is very useful and worth considering in further programming activities.

Furthermore, the users have to define the scope of the created variable, which leads them to understanding the meanings of global and local variables and demonstrates the difference between these two (Fig. 8).

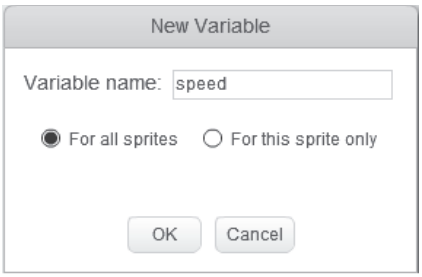

Figure 8. A variable declaration in Scratch

For example, if a sprite is active, only global and (its own) local variables can be used. Thereby students obtain the concept of the variable scope.

G. *Structured programming*

Structured programming is the most preferred approach in building programs. Students are encouraged to create a clear and structured program code. The ability to split a big task into smaller pieces plays an important role.

The majority of algorithmic languages support the definition of subroutines and functions, used in creating the code for the pieces of the project. One of the main methods of transferring data to subroutines is using the parameters. Experience shows that this is the most confusing topic for a beginner.

The new version of Scratch – Scratch 2.0 provides us with a perfect opportunity to make this issue easier. Learners can create and use their own Scratch blocks, where the definition of parameters is included (Fig. 9).

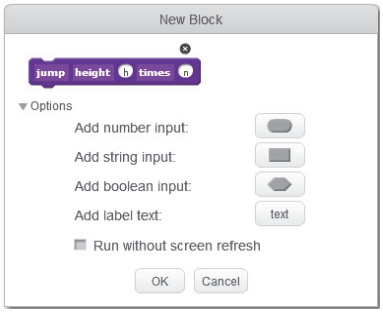

Figure 9. The user's block creation

Students learn to create a clear structure in their project. They divide their task into logical parts and create necessary user blocks, providing them with parameters. Now the main script can use standard and user-defined blocks, transferring the necessary data by means of parameters. Figure 10 shows the definition and calling of the user-defined blocks.

The issues reviewed here are very useful in the process of understanding modern concepts in building software applications and, hopefully, help students in their future study and professional work.

The next step is to proceed with more complicated tasks in other programming systems. Practicing with Scratch tools makes this function easier.

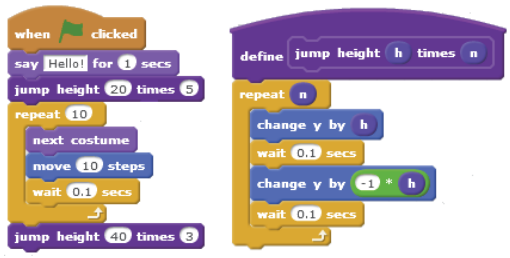

Figure 10. The main script and the user-defined block

V. SCRATCH + PYTHON

Python supports structured programming and procedural styles. In addition, Python does not require declaration of 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 the Python code to be executed 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 themselves. On the other hand, this 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 can 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 the creation of classes and use only some of the existing objects. Therefore, the object-oriented approach is not taught and is limited to the possibilities provided by Scratch only.

VI. SCRATCH + VISUAL BASIC FOR APPLICATIONS

According to our approach, Visual Basic For Applications (VBA) in MS Office applications and Scratch have a lot in common. An Excel worksheet is a big canvas, onto which a number of graphical objects can be placed. There are a number of VBA methods and functions which have unambiguous equivalents in the form of Scratch blocks (e.g. set colour). We provide beginners with ready-made procedures, to be used as "black-boxes". For example, we have procedures, that correspond to Scratch blocks such as "wait", "move", "glide", "touching" etc. The primary control statements of VBA and Scratch are closely akin. All this enables a faster transition to more sophisticated tasks we undertake in VBA.

VBA, as well as Python, supports structured programming and provides the programmer with many built-in facilities. The text editing and debugging tools for the program 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 out that using MS Excel for application creation is the easiest and most understandable way to practice the usage of 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 applied. 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 textbooks [11], [12].

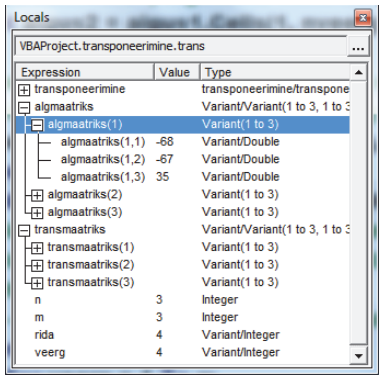

Figure 11. The Locals Window in VBA

VII. THE TEACHING METHODOLOGY

As mentioned above, we build UML (Unified Modelling Language) activity diagrams to describe algorithms in complicated tasks. We have been using a verbal description and pseudo code as well. Now, when we create Scratch projects as an introduction to programming, we can also use its scripts to visualize, formulate and describe the problem.

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 program code according to the diagrams. Later on, students have to create the models themselves.

It has to be mentioned that according to tests [13] most of our students are visual learners [14], [15]. For them it is very important to see "how it works". Scratch, with its elements of attractiveness, helps those students to understand the main idea of creating an application.

VBA already has a built-in visualising tool: students can follow the code execution using the Locals Window (Fig.11).

It automatically displays all the names of the declared variables in the current procedure, their types and their values. When the Locals Window is visible, it is automatically updated every time – students can see and check each step and its result in their program.

Python does not have such an opportunity, but still needs to be visualised. We show students the Online Python Tutor [16]. Using the visualizing tool, the students can follow each step of their code and check the values and types of the variables, as well as the order of the operators during the execution. It has to be noted that this tool has some drawbacks, as it does not support the Python graphics, time functions and work with files. However, for the beginners in programming it gives the understanding of the code execution (Fig. 12)

Moreover, for our students we create short videos about the main terms, such as iterations, branching and the execution of the processes. It should be mentioned that we surely use sound and voice records in these videos to explain the complicated moments. In our work with educational visions we follow the ideas of Khan Academy [17].

In addition, we always try to provide students´ applications with a similar content. If a student has created a model in UML and the same application in Scratch, it is easier for him to "translate" it into the VBA or Python. Thus, if a learner understands the content of the model and the algorithm, it is easier also to understand the syntax of any language.

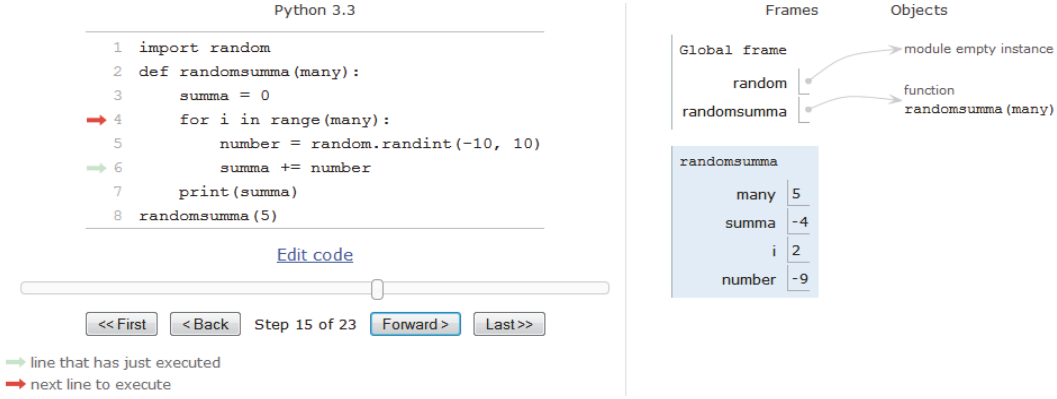

Figure 12. Python online visualizing tool

The following example is about solving a typical textbook task where the program generates a random number and asks the user to guess it. Here we see the steps of solving the task: UML activity diagram (Fig. 13), the script from Scratch project implementing the behaviour of the cat (Fig. 14) and finally the program code in Python (Fig. 15) and VBA (Fig. 16).

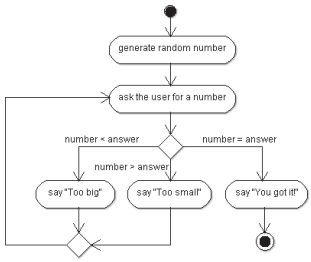

Figure. 13. UML Activity diagram

Figure. 14. Scratch project

```
from random import randint
number=randint(1,100)
while True:
    answer=int(input('Guess my number! '))
    if number==answer:
        break
    elif number>answer:
        print('Too small')
    else:
        print('Too big')
print('You got it!')
```

Figure. 15. Python code

```
Sub GuessTheNumber()
Dim answer As Integer
Randomize
Number = Int(100 * Rnd + 1)
Do
    answer = InputBox("Guess my number!")
    If Number > answer Then
        MsgBox "Too small"
    ElseIf Number < answer Then
        MsgBox "Too big"
    End If
Loop Until Number = answer
MsgBox "You got it!"
End Sub
```

Figure. 16. VBA code

VIII. CONCLUSIONS

Based on the above said, it should be concluded that in the Informatics course for the first year non-IT students we focus mostly on the model and algorithm, rather than teaching syntax and coding techniques. Visualized tools like Scratch are a good possibility to understand the main concepts of object-oriented approach and make it easier to write a programming code in any language when these concepts are clear. Similar ideas about Scratch and Python can be observed in [18].

In our course development, we try to keep up with the times and trends in computer education as [19], [20].

REFERENCES

[1] A. Robins, J. Rountree, & N. Rountree, 2003. Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 13(2), pp. 137-172. http://dx.doi.org/10.1076/csed.13.2.137.14200

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

[3] CSTA K–12 Computer Science Standards. 2011. Available at: http://csta.acm.org/Curriculum/sub/K12Standards.html

[4] AP Computer Science Principles, 2011-2016. Available at: https://advancesinap.collegeboard.org/stem/computer-science-principles

[5] CSTA Computational Thinking Task Force. Available at: http://csta.acm.org/Curriculum/sub/CompThinking.html

[6] UK. The Royal Society. „ Shut down or restart? “ The way forward for computing in UK schools. Retrieved from: https://royalsociety.org/~/media/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf

[7] UK. Computing in the national curriculum: a guide for secondary teachers. Retrieved from: http://www.computingatschool.org.uk/data/uploads/cas_secondary.pdf

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

[9] Snap! Available at: https://snap.berkeley.edu/

[10] Blocky. Google Developers. Available at: https://developers.google.com/blockly/

[11] I. Amitan, J. Vilipõld, MS Excel rakenduste põhielemendid. Tallinn: Tallinna Tehnikaülikooli Kirjastus, 2000

[12] J. Vilipõld, MS Excel arendussüsteem Visual Basic. Tallinn: Tallinna Tehnikaülikooli Kirjastus, 2000

[13] B. A. Soloman, and R. M. Felder, (n. d.). Index of Learning Styles Questionnaire. Retrieved from: http://www.engr.ncsu.edu/learningstyles/ilsweb.html

[14] O. Mironova, T. Rüütmann, I. Amitan, J. Vilipõld, M. Saar, ”Computer Science E-Courses for Students with Different Learning Styles”, in: Annals of Computer Science and Information

Systems: Federated Conference on Computer Science and Information System, Kraków, 2013, pp. 735 - 738.

[15] O. Mironova, I. Amitan, J. Vendelin, M. Saar, T. Rüütmann, "Strategies for the Individualization of an Informatics Course", in: Annals of Computer Science and Information Systems: Federated Conference on Computer Science and Information Systems, Warsaw, 2014, pp. 835 - 840. http://dx.doi.org/10.15439/2014f259

[16] Ph. Guo. Online Python Tutor. Available at: http://www.pythontutor.com/

[17] Khan Academy. Available at: https://www.khanacademy.org

[18] C. Vorderman, Computer Coding for Kids - A Unique Step-by-step Visual Guide, From Binary Code to Building Games. London, DK Children, 2014.

[19] Exploring Computer Science. Retrieved from: http://www.exploringcs.org/

[20] National Science Foundation. Retrieved from: http://www.nsf.gov/

AUTHORS

O. Mironova is with the Tallinn University of Technology, Tallinn, Estonia as a lecturer (e-mail: olga.mironova@ttu.ee).

I. Amitan is with the Tallinn University of Technology, Tallinn, Estonia (e-mail: irina.amitan@ttu.ee).

J. Vendelin is with the Tallinn University of Technology, Tallinn, Estonia as a lecturer (e-mail: jelena.vendelin@ttu.ee).

J. Vilipõld is with the Tallinn University of Technology, Tallinn, Estonia as an associate professor emeritus (e-mail: juri.vilipold@ttu.ee).

M. Saar is with the Tallinn University of Technology, Tallinn, Estonia as an educational technologist (e-mail: merike.saar@ttu.ee).

Submitted, 19 May 2015. Published as resubmitted by the authors on 08 October 2015.

Appendix E

Mironova, O.; Amitan, I.; Vilipõld, J.; Saar, M. (2016). Active learning methods in programming for non-IT students. Proceedings of International Conference on e-Learning 2016: 10th International Conference on e-Learning; Funchal, Madeira, Portugal; 1 – 3 July 2016. IADIS Press, 239−242.

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

Appendix F

Course code: IDK0021

Course title: Informatics I

ECTS credits: 4.00

Assessment form: Pass/fail assessment

Language: Estonian, English

Teaching semester: autumn

Course aims: "To develop logical, systematic and algorithmic reasoning skills and to further the ability of investigating in a systematic way the problems and tasks at hand.
" To teach the fundamentals of the principles, methods and tools for creating applications, and to equip one with the skills of creating applications for processing economic data.

Brief description of the course: The topic of the course is application development in the environment of general-purpose application software. During the course the main principles of application creation as well as its methods, tools and principal phases (problem specification, analysis, design, and realization) are considered.
Central part of the course is devoted to creating applications for processing and analyzing data tables (e.g., Excel) - its common principles, methods and tools. The topics covered include data types and organization, expressions, formulas, named ranges, structure of tables, graphs, data sorting, queries, generation of total tables, communication between tables, and the creation and use of data models.
Basics of economic and statistical models are considered.

Learning outcomes in the course: " Acquires the foundations of independent working skills.
" Can, in a logical and argued manner, justify the feasibility of tools and methods chosen for problem-solving.
" Has basic knowledge and skills for describing and modelling data and processes.
" Is familiar with the principles, methods and tools for creating documents.
" Is familiar and can use basic knowledge for specifying, modelling, and designing documents.
" Is familiar with common principles, methods and tools for creating applications and can use them for processing and analyzing data tables.
" Has and can use basic knowledge of economic and statistical models.

Type of assessment: To receive the credits for the course, 3-4 homework assignments have to be submitted and defended and 1-2 current assignments or tests have to be passed. The number of homework assignments, current assignments and tests, as well as their content, are specified in the beginning of the semester.
In order to receive credits, all homework assignments have to be defended and all current assignments and tests have to be accomplished. The evaluation of homework - passed or failed - is given based on the defence. In order to reach the threshold, the student's knowledge and skills on the topic of the respective assignment have to be at least 60%. There will be an additional test on the topic of homework assignments, current assignments and tests that have not been handed in on time, or have not been defended.

Study literature: I. Amitan, J. Vilipõld. MS Excel. Rakenduste loomise põhielemendid. Tallinn, TTÜ
J. Vilipõld. MS Excel. Rakenduste arendussüsteem Visual Basic. Tallinn. TTÜ
E-õppele orienteeritud materjalid Web_is: e-õpikud, ekraanivisioonid, harjutus

Prerequisite(s):

Work load:

	Full-time studies	**Distance learning**
Lectures	1.0	0.0
Practices	2.0	26.0
Exercises	0.0	0.0

Independent study: Independent studying is done in the form of working with the theoretical materials and preparing the homework assignments. The amount of work in stationary form - 48 hours, in non-stationary form - 80 hours.

Confirmed: 11.05.2009

Study programmes that contain the course: HAAB02/09, HAAB02/11, HAJB08/09, HAJB08/12, HAJB08/14, HAKB02/09, HAKB02/10, HAKB02/14, TAAB02/09, TAAB02/15, TABB02/09, TASB08/09, TASB08/10, TASB08/12, TASB08/13, TASB08/14, TASB08/15, TASB08/16

Appendix G

Course code: IDK0022

Course title: Informatics II

ECTS credits: 4.00

Assessment form: Graded assessment

Language: Estonian, English

Teaching semester: spring

Course aims: To develop logical, systematic and algorithmic reasoning skills and to further the ability of investigating in a systematic way the problems and tasks at hand.
To give the fundamental knowledge and basic skills of object-oriented programming and the use of the VBA development tools of mainstream application software (e.g., Excel) for creating applications.

Brief description of the course: The course addresses application creation in the VBA development environment using MS Excel. Basic principles, methods, and means of application creation and application lifecycle stages (specification, analysis, design, and realization) are covered. Object-oriented (OO) modelling and the UML language are important methods and means covered by the course. In the course of the study the following skills and knowledge are further developed: the use of objects and their properties, methods, and events; the control of program flow (choices and repetitions, parallel processes); multi-procedural programs, the use of parameters, arguments, and joint data for organizing data flow between procedures, the use of tables and arrays in programs, and the composition and presentation of algorithms. The main kinds of MS Excel applications are considered, as well as the class diagram, the referencing of objects and their properties and methods and the use of events and event procedures.
The nature of graphical objects, their properties, and methods are examined. Emphasis is put on processing arrays and dynamic tables. Their creation, editing, and linking are considered, as well as the programmatic creation and alteration of diagrams and graphs based on them. Separately are treated user forms.

Learning outcomes in the course: Acquires the foundations of problem analysis and system modelling
Can analyze relations between systems and objects and provide rationale for the algorithms and methods applied.
Is familiar with the nature of data and objects and can specify and use them in programs.
Is familiar with and can describe using VBA and UML activity diagrams the

main activities occurring in programs and algorithms.
Can organize data communication between programs and worksheet tables and create and design user forms.
Can compose programs consisting of multiple procedures and organize data flow between them.
Can use graphical objects in programs and develop in VBA their movements and animation and drawings and schemes.
Can process typical tables and arrays of economic data by using VBA programs.

Type of assessment: To receive credits for the course, 3-4 homework assignments have to be submitted and defended and current assignments or tests have to be passed. The number of homework assignments, current assignments and tests, as well as their content, are specified in the beginning of the semester. In order to fulfill the prerequisites for the subject, all homework assignments have to be defended and a given number of points from tests and running assignments has to be received. The evaluation of homework - passed or failed - is given based on the defense. To reach the threshold, the knowledge and skills on the topic of the respective assignment or test have to be at least 60%.
In case of timely submission and successful defending of homework assignments and running assignments there is an opportunity of the possibility of determining the result in case of a very successful defending.

Study literature: Vilipõld, J. MS Excel. Rakenduste arendussüsteem Visual Basic. Tallinn. TTÜ
E-õppele orienteeritud materjalid Web'is: e-õpikud, ekraanivisioonid, harjutusvihikud, näited ja demod.

Prerequisite(s): IDK0021

Work load:

	Full-time studies	**Distance learning**
Lectures	1.0	0.0
Practices	2.0	26.0
Exercises	0.0	0.0

Independent study: Independent studying is performed in the form of working with the theoretical materials and preparing the homework assignments. The amount of work in stationary form - 48 hours, in non-stationary form - 80 hours.

Confirmed: 11.05.2009

Study programmes that contain the course: HAAB02/09, HAAB02/11, HAJB08/09, HAJB08/12, HAKB02/09, HAKB02/10, HAKB02/14, TAAB02/09, TAAB02/15, TABB02/09, TASB08/09, TASB08/14, TASB08/15, TASB08/16

Appendix H

Course code: IDK0091

Course title: Informatics I (engineering)

ECTS credits: 4.00

Assessment form: Pass/fail assessment

Language: Estonian, Russian

Teaching semester: autumn

Course aims: To develop logical and analytical reasoning skills and to further the ability of investigating the problems and tasks at hand, in a systematic way. o teach the fundamentals of application creation principles, methods and tools, and to equip one with the skills of creating applications in a spreadsheet environment.

Brief description of the course: The main topic of the course is application development in the spreadsheet environment.
In the beginning of the course the main principles of application creation as well as its methods, tools and principal phases are considered. Object-oriented (OO) modelling and the UML language receive central attention.
The environment is considered according to OO principles. While considering data, formulas and functions, the emphasis is first and foremost put on problems and applications arising in the field of engineering: mathematical formulas with complex structure and with different functions, logical expressions and conditional formulas etc.
The use of tables has a central place in the course. The typical structures of the tables; the use of names and formulas in the tables; validation; the means for organizing tables, presenting queries, calculating summaries and totals are considered. Students also work with multiple linked tables, and data models resembling structures used in databases are described.
A part of the course is also an introduction to programming and the development system Visual Basic for Applications (VBA). Different types of programs and procedures are considered, also the use of simpler objects.

Learning outcomes in the course: - Acquires the foundations of independent working skills.
- Can, in a logical and argued manner, justify the feasibility of tools and methods chosen for problem-solving.
- Is familiar with the principles of creating applications as well as the methods and tools used for it, and with the principal phases of the development process.
- Is familiar with the capabilities of mainstream application software and environments and can use these (esp. spreadsheet software) efficiently to create applications.

- Can use different types of data, also formulas, expressions and internal functions of spreadsheet software (esp. Excel) in order to solve problems arising in the field of engineering.
- Can create applications by using spreadsheet programs consisting of multiple linked tables that use formulas, validation, and lookup functions as well as table processing and data analysis tools.
- Is familiar with the principles of programmatic control, has a general idea of the nature of processes occurring in programs and can create simpler programs in Visual Basic for Applications (VBA).

Type of assessment: To receive credits for the course, 3-4 homework assignments have to be submitted and defended and 3-4 current assignments or tests have to be passed. The number of homework assignments, current assignments and tests, as well as their content, are specified in the beginning of the semester.
For receiving credits, all homework assignments have to be defended and a given number of points from tests and running assignments has to be received. The evaluation of homework - passed or not passed - is given based on the defense. In order to reach the threshold, the knowledge and skills on the topic of the respective assignment or test have to be at least 60%.
There will be an additional test on homeworks, assignments and tests that have not been handed in on time or have not been defended.

Study literature: 1. I. Amitan, J. Vilipõld. MS Excel. Rakenduste loomise põhielemendid. Tallinn, TTÜ
2. J. Vilipõld. MS Excel. Rakenduste arendussüsteem Visual Basic. Tallinn. TTÜ
3. E-õppele orienteeritud materjalid Web_is: e-õpikud, ekraanivisioonid jm.

Prerequisite(s):

Work load:

	Full-time studies	**Distance learning**
Lectures	1.0	0.0
Practices	2.0	16.0
Exercises	0.0	0.0

Independent study: Independent studying is accomplished in the form of working with the theoretical materials and preparing the homework assignments. The amount of work in stationary form is 48 hours and in non-stationary form - 80 hours.

Confirmed: 28.11.2008

Study programmes that contain the course: AAAB02/09, AAGB02/09, AAVB02/09, EAEI02/09, EAEI02/15, EAKI02/09, EAKI02/15, EALB02/09, EALB02/12, EALB02/14, EATI02/09, EATI02/15, YAEB14/15, YAFB02/09, YAFB02/16, YAGB02/09, YAGB02/11, YAMB11/11, YASB02/09

Appendix I

Course code: IDK0092

Course title: Informatics II (engineering)

ECTS credits: 4.00

Assessment form: Graded assessment

Language: Estonian, Russian

Teaching semester: spring

Course aims: To develop logical, analytical and algorithmic reasoning skills and to further the ability of investigating in a systematic way the problems and tasks at hand.
To give the fundamental knowledge and basic skills of object-oriented programming and the use of the development tools of mainstream application software (esp. VBA) for creating applications.

Brief description of the course: In the course of the study the following programming related skills and knowledge are furthered and made more specific in an object oriented environment: the use of objects and their properties, methods, and events; the control of program flow (choices and repetitions, parallel processes); multi-procedural programs, the use of parameters, arguments, and joint data for organizing data flow between procedures; the use of tables and arrays in programs; the composition and presentation of algorithms.
The main kinds of MS Excel applications are considered, as well as the class diagram, the referencing of objects and their properties and methods and the use of events and event procedures.
The nature of graphical objects, their properties, and methods are thoroughly examined. Topics of computer graphics are introduced: computer screen units, the use of different coordinate systems, coordinate conversion, drawing of scaled figures and schemes, the programming of movement of graphical objects and animation, etc.
Emphasis is put on processing arrays and dynamic tables. Their creation, editing, and linking are considered, as well as the programmatic creation and alteration of diagrams and graphs based on them. Programs studying functions and depicting them graphically are considered.

Learning outcomes in the course: - Acquires the foundations of problem analysis and system modelling.
- Can analyze relations between systems and objects and provide 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 can describe using VBA and UML activity diagrams main activities occurring in programs and algorithms.
- Is familiar with the nature and main concepts of object-oriented programming and can use object class diagrams.
- Can compose programs consisting of multiple procedures and organize data flow between them.
- Can use graphical objects in programs; develop scaled drawings and schemes, movements, and animation in VBA.
- Can process tables and arrays by using VBA programs.

Type of assessment: To receive credits for the course, 3-4 homework assignments have to be submitted and defended and 3-4 current assignments or tests have to be passed. The number of homework assignments, current assignments and tests, as well as their content, are specified in the beginning of the semester.
In order to fulfil the prerequisites for the exam, all homework assignments have to be defended and a given number of points from tests and running assignments has to be received. The evaluation of homework - passed or failed - is given based on the defense. To reach the threshold, the knowledge and skills on the topic of the respective assignment or test have to be at least 60%.
the possibility of determining the examination result in case of a very successful defending.

Study literature: 1. J. Vilipõld. MS Excel. Rakenduste arendussüsteem Visual Basic. Tallinn, TTÜ.
2. E-õppele orienteeritud materjalid Webis: e-õpikud, ekraanivisioonid, harjutusvihikud, näited ja demod.

Prerequisite(s): IDK0091

Work load:

	Full-time studies	**Distance learning**
Lectures	1.0	0.0
Practices	2.0	16.0
Exercises	0.0	0.0

Independent study: Independent studying is performed by working with the theoretical materials and preparing the homework assignments. The amount of work in stationary form - 48 hours, in non-stationary form - 80 hours.

Confirmed: 11.12.2008

Study programmes that contain the course: AAAB02/09, AAGB02/09, AAVB02/09, EAEI02/09, EAEI02/15, EAKI02/09, EAKI02/15, EALB02/09, EALB02/12, EALB02/14, EATI02/09, EATI02/15, MAHB02/09, MASB02/09, MASB02/15, MATB02/09, YAEB14/15, YAFB02/09, YAMB11/11

Appendix J

Assestment method and assestment criteria in Computer Science Basics cours

ASSESSMENT METHOD	ASSESSMENT CRITERIA
Written homework (all learning outcomes)	***Assessment criteria for written homework***. *Each student will receive 3-4 homework assignments (comprising of multiple topics). The description of each of the problems is posted and the due date is communicated in a webbased environment.* *Homeworks are assessed as either being passed or failed (P/F).* *The contents of homework should correspond to the problem description and homework should be handed in by the due date.* *The acceptance of homework is a prerequisite for examination accession.* *Homework, that is handed in on time, will be accepted, if:* - *all the assignments have been solved* - *none of the parts of the solution contain important errors nor errors in principle;)* - *the homework is presented correctly and according to the requirements* *A homework that has not been accepted has to be corrected and resubmitted.* *An additional assessment will have to be passed on the topic of homeworks that have been handed in after the due date.* *All homeworks have to be defended orally during the examination – a process that takes place even in the case the homeworks contain no important errors.*
On-site assignments (all learning outcomes)	***Assessment criteria for on-site assignments.*** *Each student has to hand pass 1-2 on-site assignments.* *Fullfilled assignments form an additional prerequisite for passing the course.* *Assignments are assessed as either being passed or failed (P/F).* *An assignment that has not been passed has to be rectified.*
Self-assesment tests (all learning outcomes)	***Assessment criterion for self-assesment tests.*** *All students will receive 1-3 self-assesment tests on the topics studied.* *The purpose of the tests is for the student to self-validate his/her proficiency.* *Tests are assessed as either being passed or failed (P/F).* *A test that has not been passed has to be rectified.*
Lab work (all learning outcomes)	*All lab work solutions have to be prepared, including the self-assesment tests.*
Written credits for the course (all learning outcomes)	***Assessment criterion for written credits.*** *In order to receive prerequisite for examination accession for the course, students have to:* - *solve all lab work problems;* - *defend accepted homeworks, on-site assignments and the self-assesment tests;* - *be able to answer additional questions that are related to the*

	material at hand. *The grades given for the credits are as follows:* *„0“ – less than 50 points* *„1“ – 51- 60 points* *„2“ – 61-70 points* *„3“ – 71- 80 points* *„4“ – 81-90 points* *„5“ – 91 and more points*

Appendix K

Table 1 Reference group's results

September 2013		September 2014		January 2014		January 2015	
x	*f*	*x*	*f*	*x*	*f*	*x*	*f*
17	1	12	1	59	1	46	1
22	1	14	1	60	5	50	1
23	2	17	2	62	2	54	2
25	2	18	1	63	1	56	4
28	1	19	6	64	3	57	1
30	1	21	4	65	2	60	1
32	1	22	1	66	5	62	1
33	2	23	1	67	2	63	2
34	1	24	1	68	3	64	1
36	2	26	1	69	2	65	4
38	2	27	1	70	3	66	1
39	1	28	3	71	2	67	3
40	2	29	3	72	5	68	2
41	1	31	2	73	2	69	2
42	3	32	6	74	3	70	1
43	1	34	5	75	1	71	3
44	6	35	2	76	4	72	1
46	6	37	3	77	1	74	1

47	2	42	2	78	7	75	8
48	3	43	3	79	4	76	7
50	4	44	1	80	1	77	1
51	3	45	4	81	1	78	1
52	1	50	1	82	3	79	2
53	2	52	1	84	6	80	2
54	4	53	1	86	1	81	2
55	6	54	4	87	1	82	1
56	2	56	1	91	1	85	1
57	2	60	1	92	3	86	2
58	1	61	2	94	1	87	7
59	6	62	1	96	4	88	1
60	2	64	1	98	2	96	1
61	3	65	1	99	1	98	2
63	1	67	1	100	6	99	1
64	1	69	1			100	4
65	2	73	1				
67	1	74	1				
68	1	76	2				
69	2	82	1				
74	2						
78	1						

79	1						

Table 2 Test group's results

September 2013		September 2014		January 2014		January 2015	
x	*f*	*x*	*f*	*x*	*f*	*x*	*f*
20	1	15	1	71	1	56	1
23	1	17	3	75	1	67	1
24	1	19	2	76	1	68	1
27	1	21	4	77	2	69	1
28	3	22	5	78	1	72	2
30	1	23	1	79	4	76	2
32	1	24	3	80	4	78	2
33	1	25	3	81	4	79	6
34	2	26	1	82	2	80	1
35	2	27	4	83	1	81	1
36	1	29	2	85	1	82	1
37	1	31	4	86	4	83	1
38	2	32	2	87	3	84	1
39	1	34	3	88	3	85	3
40	6	36	1	89	5	86	2
42	3	38	2	90	7	87	2

43	2	39	2	91	4	88	3
44	8	41	1	92	6	89	5
46	4	42	1	93	2	91	2
47	3	43	6	94	3	92	1
48	7	45	3	95	1	93	3
49	2	47	1	96	2	94	5
50	1	50	1	98	16	95	4
51	2	51	2	100	11	96	4
52	4	52	1			97	4
53	2	53	2			98	8
54	3	54	1			99	2
55	7	56	1			100	6
56	2	60	1				
57	1	61	1				
58	1	65	1				
59	2	67	2				
61	3	69	1				
63	4	71	1				
69	2	72	1				
71	1	74	1				
		75	1				
		78	1				

		83	1				

Table 3 Theoretical t values

Degrees of freedom	Probability			
	0,1	0,05	0,01	0,001
∞	1,65	1,96	2,58	3,29

Appendix L

Table 4 Exams results during the experiment

Academic year	Number of students who did not attend the exam	Grades						Total number of students	Number of students who attend the exam	Average grade
		0	1	2	3	4	5			
2010/11	13	7	15	35	98	7	0	175	162	2,5
2011/12	10	6	13	25	99	19	2	174	164	2,7
2012/13	5	7	3	17	113	19	15	179	174	3,0
2013/14	2	0	0	7	37	89	43	178	176	4,0
2014/15	2	0	0	3	12	84	49	150	148	4,2
2015/16	1	0	0	2	9	46	50	108	107	4,3

CURRICULUM VITAE

Personal data

Name: Olga Mironova

Date of birth: 07.09.1978

Place of birth: Tallinn, Estonia

Citizenship: Estonian

Contact data

Address: Akadeemia tee 15a, 12618 Tallinn, Estonia

Phone: +372 5210042

E-mail: olga.mironova@ttu.ee

Education

2005–...Tallinn University of Technology, PhD

2002–2004 Tallinn University of Technology, M. Sc

1996–2001 Tallinn Pedagogical University, B. Sc, M. Sc

1993–1996 Juhkentali Secondary School

1985–1993 Suurpea primary school

Language competence

English	Average
Estonian	Fluent
Russian	Mother tongue

Professional employment

01.01.2017–... Tallinn University of Technology, School of Information Technologies, Department of Software Science, Lecturer (1,00)

2015–31.12.2016 Tallinn University of Technology, Faculty of Information Technology, Department of Informatics, Chair of Software Engineering, Lecturer (1,00)

2008–2015 Tallinn University of Technology, Faculty of Information Technology, Department of Informatics, Chair of Software Engineering, Assistant (1,00)

2006–2008 Tallinn University of Technology, Faculty of Information Technology, Department of Informatics, Chair of Software Engineering, Maternity leave (1,00)

2004–2006 Tallinn University of Technology, Faculty of Information Technology, Department of Informatics, Chair of Software Engineering, Assistant (1,00)

2002–2004 Tallinn University of Technology, Assistant (1,00)

2001–2004 Tallinn Downtown Secondary School, Teacher (1,00)

ELULOOKIRJELDUS

Isikuandmed

Nimi: Olga Mironova

Sünniaeg: 07.09.1978

Sünnikoht: Tallinn, Eesti

Kodakondsus: Eesti

Kontaktandmed

Aadress: Akadeemia tee 15a, 12618 Tallinn, Eesti

Telefon: +372 5210042

E-mail: olga.mironova@ttu.ee

Hariduskäik

2005–...Tallinna Tehnikaülikool, doktoriõppe

2002–2004 Tallinna Tehnikaülikool, magistiõppe

1996–2001 Tallinna Pedagoogikaülikool, bakalaureuse- ja magistriõppe

1993–1996 Tallinna Juhkentali Gümnaasium

1985–1993 Suurpea põhikool

Keelteoskus

Inglise keel Kesktase

Eesti keel Kõrgtase

Vene keel Emakeel

Teenistuskäik

01.01.2017–... Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Tarkvarateaduse instituut, Lektor (1,00)

2015–31.12.2016 Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Informaatikainstituut, Tarkvaratehnika õppetool, Lektor (1,00)

2008–2015 Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Informaatikainstituut, Tarkvaratehnika õppetool, Assistent (1,00)

2006–2008 Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Informaatikainstituut, Tarkvaratehnika õppetool, Lapsepuhkus (1,00)

2004–2006 Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Informaatikainstituut, Tarkvaratehnika õppetool, Assistent (1,00)

2002–2004 Tallinna Tehnikaülikool, tunnitasuline õppejõud (1,00)

2001–2004 Tallinna Kesklinna Vene Gümnaasium, Õpetaja (1,00)

DISSERTATIONS DEFENDED AT TALLINN UNIVERSITY OF TECHNOLOGY ON *INFORMATICS AND SYSTEM ENGINEERING*

1. **Lea Elmik**. Informational Modelling of a Communication Office. 1992.

2. **Kalle Tammemäe**. Control Intensive Digital System Synthesis. 1997.

3. **Eerik Lossmann**. Complex Signal Classification Algorithms, Based on the Third-Order Statistical Models. 1999.

4. **Kaido Kikkas**. Using the Internet in Rehabilitation of People with Mobility Impairments – Case Studies and Views from Estonia. 1999.

5. **Nazmun Nahar**. Global Electronic Commerce Process: Business-to-Business. 1999.

6. **Jevgeni Riipulk**. Microwave Radiometry for Medical Applications. 2000.

7. **Alar Kuusik**. Compact Smart Home Systems: Design and Verification of Cost Effective Hardware Solutions. 2001.

8. **Jaan Raik**. Hierarchical Test Generation for Digital Circuits Represented by Decision Diagrams. 2001.

9. **Andri Riid**. Transparent Fuzzy Systems: Model and Control. 2002.

10. **Marina Brik**. Investigation and Development of Test Generation Methods for Control Part of Digital Systems. 2002.

11. **Raul Land**. Synchronous Approximation and Processing of Sampled Data Signals. 2002.

12. **Ants Ronk**. An Extended Block-Adaptive Fourier Analyser for Analysis and Reproduction of Periodic Components of Band-Limited Discrete-Time Signals. 2002.

13. **Toivo Paavle**. System Level Modeling of the Phase Locked Loops: Behavioral Analysis and Parameterization. 2003.

14. **Irina Astrova**. On Integration of Object-Oriented Applications with Relational Databases. 2003.

15. **Kuldar Taveter**. A Multi-Perspective Methodology for Agent-Oriented Business Modelling and Simulation. 2004.

16. **Taivo Kangilaski**. Eesti Energia käiduhaldussüsteem. 2004.

17. **Artur Jutman**. Selected Issues of Modeling, Verification and Testing of Digital Systems. 2004.

18. **Ander Tenno**. Simulation and Estimation of Electro-Chemical Processes in Maintenance-Free Batteries with Fixed Electrolyte. 2004.

19. **Oleg Korolkov**. Formation of Diffusion Welded Al Contacts to Semiconductor Silicon. 2004.

20. **Risto Vaarandi**. Tools and Techniques for Event Log Analysis. 2005.

21. **Marko Koort**. Transmitter Power Control in Wireless Communication Systems. 2005.

22. **Raul Savimaa**. Modelling Emergent Behaviour of Organizations. Time-Aware, UML and Agent Based Approach. 2005.

23. **Raido Kurel**. Investigation of Electrical Characteristics of SiC Based Complementary JBS Structures. 2005.

24. **Rainer Taniloo**. Ökonoomsete negatiivse diferentsiaaltakistusega astmete ja elementide disainimine ja optimeerimine. 2005.

25. **Pauli Lallo**. Adaptive Secure Data Transmission Method for OSI Level I. 2005.

26. **Deniss Kumlander**. Some Practical Algorithms to Solve the Maximum Clique Problem. 2005.

27. **Tarmo Veskioja**. Stable Marriage Problem and College Admission. 2005.

28. **Elena Fomina**. Low Power Finite State Machine Synthesis. 2005.

29. **Eero Ivask**. Digital Test in WEB-Based Environment 2006.

30. **Виктор Войтович**. Разработка технологий выращивания из жидкой фазы эпитаксиальных структур арсенида галлия с высоковольтным p-n переходом и изготовления диодов на их основе. 2006.

31. **Tanel Alumäe**. Methods for Estonian Large Vocabulary Speech Recognition. 2006.

32. **Erki Eessaar**. Relational and Object-Relational Database Management Systems as Platforms for Managing Softwareengineering Artefacts. 2006.

33. **Rauno Gordon**. Modelling of Cardiac Dynamics and Intracardiac Bio-impedance. 2007.

34. **Madis Listak**. A Task-Oriented Design of a Biologically Inspired Underwater Robot. 2007.

35. **Elmet Orasson**. Hybrid Built-in Self-Test. Methods and Tools for Analysis and Optimization of BIST. 2007.

36. **Eduard Petlenkov**. Neural Networks Based Identification and Control of Nonlinear Systems: ANARX Model Based Approach. 2007.

37. **Toomas Kirt**. Concept Formation in Exploratory Data Analysis: Case Studies of Linguistic and Banking Data. 2007.

38. **Juhan-Peep Ernits**. Two State Space Reduction Techniques for Explicit State Model Checking. 2007.

39. **Innar Liiv**. Pattern Discovery Using Seriation and Matrix Reordering: A Unified View, Extensions and an Application to Inventory Management. 2008.

40. **Andrei Pokatilov**. Development of National Standard for Voltage Unit Based on Solid-State References. 2008.

41. **Karin Lindroos**. Mapping Social Structures by Formal Non-Linear Information Processing Methods: Case Studies of Estonian Islands Environments. 2008.

42. **Maksim Jenihhin**. Simulation-Based Hardware Verification with High-Level Decision Diagrams. 2008.

43. **Ando Saabas**. Logics for Low-Level Code and Proof-Preserving Program Transformations. 2008.

44. **Ilja Tšahhirov**. Security Protocols Analysis in the Computational Model – Dependency Flow Graphs-Based Approach. 2008.

45. **Toomas Ruuben**. Wideband Digital Beamforming in Sonar Systems. 2009.

46. **Sergei Devadze**. Fault Simulation of Digital Systems. 2009.

47. **Andrei Krivošei**. Model Based Method for Adaptive Decomposition of the Thoracic Bio-Impedance Variations into Cardiac and Respiratory Components. 2009.

48. **Vineeth Govind**. DfT-Based External Test and Diagnosis of Mesh-like Networks on Chips. 2009.

49. **Andres Kull**. Model-Based Testing of Reactive Systems. 2009.

50. **Ants Torim**. Formal Concepts in the Theory of Monotone Systems. 2009.

51. **Erika Matsak**. Discovering Logical Constructs from Estonian Children Language. 2009.

52. **Paul Annus**. Multichannel Bioimpedance Spectroscopy: Instrumentation Methods and Design Principles. 2009.

53. **Maris Tõnso**. Computer Algebra Tools for Modelling, Analysis and Synthesis for Nonlinear Control Systems. 2010.

54. **Aivo Jürgenson**. Efficient Semantics of Parallel and Serial Models of Attack Trees. 2010.

55. **Erkki Joasoon**. The Tactile Feedback Device for Multi-Touch User Interfaces. 2010.

56. **Jürgo-Sören Preden**. Enhancing Situation – Awareness Cognition and Reasoning of Ad-Hoc Network Agents. 2010.

57. **Pavel Grigorenko**. Higher-Order Attribute Semantics of Flat Languages. 2010.

58. **Anna Rannaste**. Hierarcical Test Pattern Generation and Untestability Identification Techniques for Synchronous Sequential Circuits. 2010.

59. **Sergei Strik**. Battery Charging and Full-Featured Battery Charger Integrated Circuit for Portable Applications. 2011.

60. **Rain Ottis**. A Systematic Approach to Offensive Volunteer Cyber Militia. 2011.

61. **Natalja Sleptšuk**. Investigation of the Intermediate Layer in the Metal-Silicon Carbide Contact Obtained by Diffusion Welding. 2011.

62. **Martin Jaanus**. The Interactive Learning Environment for Mobile Laboratories. 2011.

63. **Argo Kasemaa**. Analog Front End Components for Bio-Impedance Measurement: Current Source Design and Implementation. 2011.

64. **Kenneth Geers**. Strategic Cyber Security: Evaluating Nation-State Cyber Attack Mitigation Strategies. 2011.

65. **Riina Maigre**. Composition of Web Services on Large Service Models. 2011.

66. **Helena Kruus**. Optimization of Built-in Self-Test in Digital Systems. 2011.

67. **Gunnar Piho**. Archetypes Based Techniques for Development of Domains, Requirements and Sofware. 2011.

68. **Juri Gavšin**. Intrinsic Robot Safety Through Reversibility of Actions. 2011.

69. **Dmitri Mihhailov**. Hardware Implementation of Recursive Sorting Algorithms Using Tree-like Structures and HFSM Models. 2012.

70. **Anton Tšertov**. System Modeling for Processor-Centric Test Automation. 2012.

71. **Sergei Kostin**. Self-Diagnosis in Digital Systems. 2012.

72. **Mihkel Tagel**. System-Level Design of Timing-Sensitive Network-on-Chip Based Dependable Systems. 2012.

73. **Juri Belikov**. Polynomial Methods for Nonlinear Control Systems. 2012.

74. **Kristina Vassiljeva**. Restricted Connectivity Neural Networks based Identification for Control. 2012.

75. **Tarmo Robal**. Towards Adaptive Web – Analysing and Recommending Web Users` Behaviour. 2012.

76. **Anton Karputkin**. Formal Verification and Error Correction on High-Level Decision Diagrams. 2012.

77. **Vadim Kimlaychuk**. Simulations in Multi-Agent Communication System. 2012.

78. **Taavi Viilukas**. Constraints Solving Based Hierarchical Test Generation for Synchronous Sequential Circuits. 2012.

79. **Marko Kääramees**. A Symbolic Approach to Model-based Online Testing. 2012.

80. **Enar Reilent**. Whiteboard Architecture for the Multi-agent Sensor Systems. 2012.

81. **Jaan Ojarand**. Wideband Excitation Signals for Fast Impedance Spectroscopy of Biological Objects. 2012.

82. **Igor Aleksejev**. FPGA-based Embedded Virtual Instrumentation. 2013.

83. **Juri Mihhailov**. Accurate Flexible Current Measurement Method and its Realization in Power and Battery Management Integrated Circuits for Portable Applications. 2013.

84. **Tõnis Saar**. The Piezo-Electric Impedance Spectroscopy: Solutions and Applications. 2013.

85. **Ermo Täks**. An Automated Legal Content Capture and Visualisation Method. 2013.

86. **Uljana Reinsalu**. Fault Simulation and Code Coverage Analysis of RTL Designs Using High-Level Decision Diagrams. 2013.

87. **Anton Tšepurov**. Hardware Modeling for Design Verification and Debug. 2013.

88. **Ivo Müürsepp**. Robust Detectors for Cognitive Radio. 2013.

89. **Jaas Ježov**. Pressure sensitive lateral line for underwater robot. 2013.

90. **Vadim Kaparin**. Transformation of Nonlinear State Equations into Observer Form. 2013.

92. **Reeno Reeder**. Development and Optimisation of Modelling Methods and Algorithms for Terahertz Range Radiation Sources Based on Quantum Well Heterostructures. 2014.

93. **Ants Koel**. GaAs and SiC Semiconductor Materials Based Power Structures: Static and Dynamic Behavior Analysis. 2014.

94. **Jaan Übi**. Methods for Coopetition and Retention Analysis: An Application to University Management. 2014.

95. **Innokenti Sobolev**. Hyperspectral Data Processing and Interpretation in Remote Sensing Based on Laser-Induced Fluorescence Method. 2014.

96. **Jana Toompuu**. Investigation of the Specific Deep Levels in *p*-, *i*- and *n*-Regions of GaAs p^+-*pin*-n^+ Structures. 2014.

97. **Taavi Salumäe**. Flow-Sensitive Robotic Fish: From Concept to Experiments. 2015.

98. **Yar Muhammad**. A Parametric Framework for Modelling of Bioelectrical Signals. 2015.

99. **Ago Mõlder**. Image Processing Solutions for Precise Road Profile Measurement Systems. 2015.

100. **Kairit Sirts**. Non-Parametric Bayesian Models for Computational Morphology. 2015.

101. **Alina Gavrijaševa**. Coin Validation by Electromagnetic, Acoustic and Visual Features. 2015.

102. **Emiliano Pastorelli**. Analysis and 3D Visualisation of Microstructured Materials on Custom-Built Virtual Reality Environment. 2015.

103. **Asko Ristolainen**. Phantom Organs and their Applications in Robotic Surgery and Radiology Training. 2015.

104. **Aleksei Tepljakov**. Fractional-order Modeling and Control of Dynamic Systems. 2015.

105. **Ahti Lohk**. A System of Test Patterns to Check and Validate the Semantic Hierarchies of Wordnet-type Dictionaries. 2015.

106. **Hanno Hantson**. Mutation-Based Verification and Error Correction in High-Level Designs. 2015.

107. **Lin Li**. Statistical Methods for Ultrasound Image Segmentation. 2015.

108. **Aleksandr Lenin**. Reliable and Efficient Determination of the Likelihood of Rational Attacks. 2015.

109. **Maksim Gorev**. At-Speed Testing and Test Quality Evaluation for High-Performance Pipelined Systems. 2016.

110. **Mari-Anne Meister**. Electromagnetic Environment and Propagation Factors of Short-Wave Range in Estonia. 2016.

111. **Syed Saif Abrar**. Comprehensive Abstraction of VHDL RTL Cores to ESL SystemC. 2016.

112. **Arvo Kaldmäe**. Advanced Design of Nonlinear Discrete-time and Delayed Systems. 2016.

113. **Mairo Leier**. Scalable Open Platform for Reliable Medical Sensorics. 2016.

114. **Georgios Giannoukos**. Mathematical and Physical Modelling of Dynamic Electrical Impedance. 2016.

115. **Aivo Anier**. Model Based Framework for Distributed Control and Testing of Cyber-Physical Systems. 2016.

116. **Denis Firsov**. Certification of Context-Free Grammar Algorithms. 2016.

117. **Sergei Astapov**. Distributed Signal Processing for Situation Assessment in Cyber-Physical Systems. 2016.

118. **Erkki Moorits**. Embedded Software Solutions for Development of Marine Navigation Light Systems. 2016.

119. **Andres Ojamaa**. Software Technology for Cyber Security Simulations. 2016.

120. **Gert Toming**. Fluid Body Interaction of Biomimetic Underwater Robots. 2016.

121. **Kadri Umbleja**. Competence Based Learning – Framework, Implementation, Analysis and Management of Learning Process. 2017.

122. **Andres Hunt**. Application-Oriented Performance Characterization of the Ionic Polymer Transducers (IPTs). 2017.

123. **Niccolò Veltri**. A Type-Theoretical Study of Nontermination. 2017.

124. **Tauseef Ahmed**. Radio Spectrum and Power Optimization Cognitive Techniques for Wireless Body Area Networks. 2017.

125. **Andre Veski**. Agent-Based Computational Experiments in Two-Sided Matching Markets. 2017

126. **Artjom Rjabov**. Network-Based Hardware Accelerators for Parallel Data Processing. 2017.

127. **Fatih Güllü**. Conformity Analysis of E-Learning Systems at Largest Universities in Estonia and Turkey on the Basis of EES Model.

128. **Margarita Spitšakova**. Discrete Gravitational Swarm Optimization Algorithm for System Identification. 2017.