VILNIUS UNIVERSITY

JŪRATĖ URBONIENĖ

DESIGN OF ADAPTIVE PROGRAMMING TEACHING TOOLS

Summary of Doctoral Dissertation
Technological Sciences, Informatics Engineering (07 T)

Vilnius, 2014

The doctoral dissertation was accomplished at Vilnius University Institute of Mathematics and Informatics in the period from 2009 to 2013.

Scientific Supervisor
Prof. Dr. Valentina Dagienė (Vilnius University, Technological Sciences, Informatics Engineering – 07 T).

The dissertation will be defended at the Council of the Scientific Field of Informatics Engineering at Vilnius University Institute of Mathematics and Informatics:

Chairman
Prof. Dr. Habil. Gintautas Dzemyda (Vilnius University, Technological Sciences, Informatics Engineering – 07 T).

Members:
Prof. Dr. Habil. Juozas Augutis (Vytautas Magnus University, Physical Sciences, Informatics – 09 P),
Prof. Dr. Habil. Marijona Barkauskaitė (Lithuanian University of Educational Sciences, Social Sciences, Education – 07 S),
Assoc. Prof. Dr. Regina Kulvietienė (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – 07 T),
Assoc. Prof. Dr. Olga Kurasova (Vilnius University, Technological Sciences, Informatics Engineering – 07 T).

Opponents:
Assoc. Prof. Dr. Vitalijus Denisovas (Klaipėda University, Technological Sciences, Informatics Engineering – 07 T),
Dr. Jevgenij Kurilov (Vilnius University, Technological Sciences, Informatics Engineering – 07 T).

The dissertation will be defended at the public session of the Scientific Council of the Scientific Field of Informatics Engineering in the auditorium number 203 at Vilnius University Institute of Mathematics and Informatics, at 1 p. m. on the 13th of June, 2014.

Address: Akademijos st. 4, LT-08663 Vilnius, Lithuania.

The summary of the doctoral dissertation was distributed on the 13th of May 2014.

A copy of the doctoral dissertation is available for review at the Library of Vilnius University.

VILNIAUS UNIVERSITETAS

JŪRATĖ URBONIENĖ

ADAPTYVIŲJŲ PROGRAMAVIMO MOKYMO PRIEMONIŲ PROJEKTAVIMAS

Daktaro disertacijos santrauka
Technologijos mokslai, informatikos inžinerija (07 T)

Vilnius, 2014 metai

Disertacija rengta 2009 – 2013 metais Vilniaus universiteto Matematikos ir informatikos institute

Mokslinė vadovė
prof. dr. Valentina Dagienė (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Disertacija ginama Vilniaus universiteto Matematikos ir informatikos instituto Informatikos inžinerijos mokslo krypties taryboje:

Pirmininkas
prof. habil. dr. Gintautas Dzemyda (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Nariai:
prof. habil. dr. Juozas Augutis (Vytauto Didžiojo universitetas, fiziniai mokslai, informatika – 09 P),
prof. habil. dr. Marijona Barkauskaitė (Lietuvos edukologijos universitetas, socialiniai mokslai, edukologija – 07 S),
doc. dr. Regina Kulvietienė (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – 07T),
doc. dr. Olga Kurasova (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Oponentai:
doc. dr. Vitalijus Denisovas (Klaipėdos universitetas, technologijos mokslai, informatikos inžinerija – 07 T),
dr. Jevgenij Kurilov (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Disertacija bus ginama viešame Informatikos inžinerijos mokslo krypties tarybos posėdyje 2014 m. birželio mėn. 13 d. 13 val. Vilniaus universiteto Matematikos ir informatikos instituto 203 auditorijoje.

Adresas: Akademijos g. 4, Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta 2014 m. gegužės mėn. 13 d.

Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje.

1. GENERAL CHARACTERISTICS OF THE DISSERTATION

1.1. SCOPE AND RELEVANCE

Teaching programming is a sufficiently acute problem not only in Lithuania, but also in many foreign countries. Many scientific articles have been written on this topic, not only one thesis was prepared and defended (Bennedsen, 2008; Caspersen, 2007; Kinnunen 2009; Sorva 2012, etc..), there are several strong research groups that engage in programming, algorithmization teaching matters (for example, Aalto, Joensuu, Technical University of Munich). Currently, personalized training (further used learning) is actively investigated in the world, and it may be associated with adaptive teaching tools – systems.

To learn how to program is not easy. The learner must first develop algorithmic thinking. The main feature of this way of thinking is the ability to express assignment into the algorithm so that it can be resolved by the computer (i.e. be able to program). What tools and techniques are used to teach algorithmic thinking? Usually tutor or teacher introduces programming language syntax, algorithms, and shows examples of programs on the display board or projector. It takes a lot of time and is inefficient. There are rarely used interactive whiteboards or projectors, where the program written by a teacher is clearly visible to all learners and they can examine it altogether. Positive impact on learning has information visualization (Ebel, Ben-Ari, 2006), but it is usually used as a teaching tool - simply the teacher presents material in a different form, using video footage.

It is emphasized that in action (completing practical tasks) human memory optimizes learning (Caspersen, 2007). Also important is the fact that quality of attention is correlated with the effectiveness of the learning - the more time learner's attention is maintained, the better the outcomes (Ebel, Ben-Ari, 2006). The modern technology, computers and the Internet provides opportunities to make learning more interesting and effective, to facilitate teacher’s work, and motivate the learner.

Teaching and learning programming (algorithmization) is important not only to correct errors or the number of solutions, but also the interaction and communication of the learner and teacher. Often the interaction takes place in absentia, through an intermediary - the computer. By learning, a lot of time is spent on how to correct errors, to learn from them, how to get answers to the concerning questions in time. For that matter, there various automatic programming and algorithmization training systems are developed (for example, TRAKLA2, VILLE, etc.). These systems teach the principles of programming and creating algorithms, but they are not friendly enough for the learner, and especially they lack adaptability.

For decades, attempt has been made to find more effective teaching and learning tools, methods and techniques. There is a growing emphasis on the importance of personalized learning - when learning objects (LO), learning activities, learning scenarios, and so on, are selected taking into account the individual characteristics of the learner. Adaptive learning systems that allow during the learning process to adapt appropriate learning contents for the learner also help to ensure individualization. At the same time, attempt is being made to adapt and use learning elements - LO, which were previously developed, which are qualitative and stored in the repository.

Adaptation of learning to individual learner’s needs in Lithuania and abroad has been long analyzed. There are many studies in which using a variety of methods -

according to the characteristics and needs of the learner (Weber, Brusilovsky, 2001; Baziukaitė and others., 2008), analyzing learner's activities in the electronic learning environment (Preidys, Žilinskienė, 2012), identifying learners learning styles (Franzoni, Assar, 2009) - are chosen or allowed for the learner to choose the most suitable learning content or learning strategy (Preidys, Žilinskienė, 2012), when in the learning environment adaptive navigation (Weber, Brusilovsky, 2001; Paramythis, Loidl-Reisinger, 2004) is maintained. Learning success depends on how maximum achievable learning objectives are – how the necessary subject competences (knowledge and skills, which are described in the learning outcomes) are obtained, and what emotions the learners have experienced in learning. Learning success is largely determined by the learning efficiency, which depends on the willingness to learn and ability to learn. The last determines also the individual learner's characteristics, which vary according to the learning style. In scientists work it is mostly dealt with learning adaptation in terms of learning styles according to Kolb, Honey - Mumford, Felder - Silverman classifications (Hawk, Shah, 2007). Meanwhile, there are just a few works in which Herrmann style classification was examined in the context of computer science education (Kannangara, 2013), but the use in learning systems was not detected.

The dissertation examines programming teaching subject material adaptation to the learner's individual qualities - his learning style by Herrmann learning style classification. To that purpose, programming teaching features, existing programming teaching environments and systems, adaptive learning systems, software agents and agent systems, students learning styles and learning material elements and their repositories and learning systems development methods were examined. In this work authors created methods are presented: method for ranking of LO types by relevance of Herrmann learning style; method for improved teaching subject curriculum compliance with the subject area of LO and Bloom taxonomy levels; method for adapting elements of programming learning materials (LO) to the Herrmann learning style of the learner. All of these methods are integrated as individual agents of multiagent software system to adaptive programming teaching tools model (further - adaptive programming teaching system - APTS). Developed model was evaluated by selected experts - the quality of the use of the model i.e. effectiveness, efficiency, productivity, usability and functionality by adapting teaching subject material to the learner has been shown.

1.2. RESEARCH OBJECT

The research object of the dissertation is adaptive programming teaching tool (APTS).

1.3. RESEARCH AIM

The aim of dissertation work - after studying the problem of programming teaching and student learning possibilities and the needs to develop a conceptual model of adaptive programming teaching system.

1.4. RESEARCH OBJECTIVES

In order to achieve the main goal, the following tasks were raised:

1 Perform comparative analysis of the software solutions used for teaching programming in relation to adaptivity.

2 Distinguished learning object types suitable for teaching programming and classify them according to the learner's learning style.
3 Create a method that allows selecting programming learning objects from learning objects repositories according to the curriculum requirements and the learner's learning style.
4 Develop a conceptual model of adaptive programming teaching system, which adapts learning materials to the learner taking into account their learning style.
5 Perform qualitative assessment of adaptive programming teaching system model's usage.

1.5. RESEARCH METHODOLOGY AND DATA ANALYSIS METHODS

For preparation of the dissertation work, the following research and data analysis methods were used:

- systematic analysis of scientific literature and research - it was used when analyzing the peculiarities of teaching programming, techniques, issues, adaptive and agent system characteristics;
- research of programming teaching tools - used while trying to figure characteristics of tools used for teaching programming in terms of adaptability;
- questionnaire - used for the interpretation of student learning needs;
- Expert assessment - used in assessing suitability of the learning object types for the learning style and the quality of use of modeled system;
- Modeling - used in describing and developing adaptive programming teaching system model.

1.6. SCIENTIFIC NOVELTY

- A method for learning object types of expert evaluation results usage for the learning object types ranking according to the Herrmann learning styles was prepared, that enables to personalize learning;
- A method for curriculum linking to the subject area of learning object and Bloom taxonomy levels was proposed;
- Method by which the programming teaching subject of learning material (learning objects) is adapted to the learner according to Herrmann learning style was prepared.

1.7. DEFENDED PROPOSITIONS

- Learning object types eligibility of learner's expert assessment for Herrmann learning style helps adapt learning objects to achieve learning results of teaching subject. This method allows you to associate leraning object types with learners' learning style and that way to personalize learning;
- Learning object compliance with the curriculum method allows you to create technology that links selected elements of a subject matterial (learning objects) to the Bloom's Taxonomy levels, and at the same time forms the formal assessment's task weights to assess professional competencies;

- Programming learning system that adapts learning objects to the learner according to his learning style, allows you to pursue competences of the subject in the program and to assess the level of achievement;
- The system described by adaptive programming teaching system model is effective, efficient, productive and usable (flexible and context conformable).

1.8. PRACTICAL IMPORTANCE

- Using the proposed model of the system, it is possible to create a system that offers the following parameters: effectiveness, efficiency, productivity and usability (flexibility and context conformity). This system can be integrated as separate module in to existing virtual learning environments like Moodle.
- The proposed model can be applied to other areas of learning, respectively distinguishing types of learning object the most suitable for learning and in a curriculum listing naming the themes, subject-specific competences and Bloom's taxonomy levels.

1.9. PUBLICATION AND APPROBATION

The results of the Doctoral thesis were published in 7 scientific publications in periodical peer-reviewed journals. The full list of publications on the topic of the Doctoral thesis is presented at the end of this Summary.

The results of the Doctoral thesis were presented via 7 presentations, given in international and national conferences and seminars as follows:

- Report "The adaptive Technologies in learning process". VIII Scientific-practical conference „Information Technology 2010: The Theory, Practice, Innovation", Alytus College, Alytus, Lithuania. 2010-05-21
- Report "Web 2.0 technology feasibility study for programming teaching". Scientific-practical conference "Studies in modern society". North Lithuania College, Šiauliai, Lithuania. 2011-03-01
- Stand report "Internet technologies and advanced programming teaching". International scientific-practical conference "Sustainable development aspects: theory and practice". Utena College, Utena, Lithuania. 14th and 15th of april, 2011.
- Report "Learning using Internet technologies". Republican scientific and practical conference "Research and Studies 2011: Theory and Practice". North Lithuania College, Šiauliai, Lithuania. 2011-05-12
- Report "The analysis of efficiency of programming teaching environments in relation to teaching (learning)". Studies in modern society. Scientific and practical conference "Studies in Modern Society". North Lithuania College, Šiauliai, Lithuania. 2012-02-23
- Report "Cooperation in Programming Learning". 12th Koli Calling International Conference on Computing Education Research. Tahko, Finland, 2012-11-15 – 2012-11-18
- Report "The Possibilities of Virtual Learning Environment Tool Usability for Programming Training". International scientific-practical conference "Innovative

Information Technologies for Science, Business and Education, IIT-2013". Vilnius business College, Vilnius, 2013-11-14 – 2013-11-16

The results of the Doctoral thesis were presented via 7 presentations, given in international doctoral consorciums as follows:

- International doctoral training "JTEL Summer School 2010 on Technology Enhanced Learning". Ohrid, FYR Macedonia. 2010-06-10
- International doctoral training "JTEL Summer School 2011 on Technology Enhanced Learning". Chania, Crete, Greece. 2011-06-01
- International doctoral training: "Informatics and Informatics Engineering Education Research: Methodologies, Strategies and Implementation". Druskininkai, Lithuania. 2011-12-02
- International doctoral training "JTEL Summer School 2012 on Technology Enhanced Learning" (Estoril, Portugal). 2012-05-22
- International doctoral training "Informatics Engineering Education Research: Methodologies, Methods and Practice". Druskininkai, Lithuania, 2012-12-03 – 2012-12-07
- 10th World Conference on Computers in Education "Learning while we are connected". International doctoral consortium. Torun, Poland, 2013-05-01 – 2013-05-05
- International doctoral training "Informatics Engineering Education Research: Methodologies, Methods and Practice". Druskininkai, Lithuania, 2013-12-03 – 2013-12-07

1.10. SCOPE AND STRUCTURE

The Doctoral dissetration consists of four chapters, general conclusions and results, references, and annexes. The work includes 170 pages of text, 76 figures, 28 tables, 6 appendices. The Doctoral thesis is written in Lithuanian.

The *first* (introductory) chapter includes the statement of the problem, its relevance, research aim and objectives, scientific novelty and practical importance, as well as work's approbation and publications.

The *second* chapter is analytical. This chapter examines programming teaching features of existing programming teaching environment and system, adaptive technology and learning systems, software agents, and Agent system, student learning styles and elements of learning material and their repositories, the learning system development models.

The *third* section describes the designing of APTS conceptual model. The system is designed as multiagent system, which uses qualitative programming LO which are relevant (selected) for the subject in the curriculum and are protected in a specialized LO repository. LO are selected and presented by rating to the learner, based on his Herrmann learning style, while appropriateness of the LO types for the learning styles is determined by using the method of expert evaluation.

The *fourth* chapter presents quality assessment of using the generated APTS model. Expert assessment method was used which allows evaluating quality of the developed system model usage by the following aspects: effectiveness, efficiency, productivity, usability and functionality.

Summary of the results and conclusions is presented at the end of the work.

2. TEACHING PROGRAMMING AND ANALYSIS OF THE ADAPTIVE SYSTEMS LITERATURE

In this section, the results of the analytical part of the dissertation (Chapter 2) are briefly presented.

2.1. Features of teaching programming

Programming teaching may be divided into two broad categories: 1) programming language syntax and mastering semantics, 2) learning of programming techniques (methods). Programming language syntax and semantics are necessary for a beginner to be able to write programs. The most important thing is not to learn the syntax structures, but to be able to apply them. Programming techniques (methods) covers the basic concepts and technical training. Learning programming while writing program is just one of the programming skills. Equally important is the ability to read and understand the texts of the programs. After all programmer spends a lot of time examining examples - written by other programs (Mannila, 2007; Blonskis, Dagienė, 2003). It is also important to develop a logical and algorithmic thinking - the ability to analyze real-life problems, to split them into smaller tasks, reason choices of solutions and to synthesize the results to give an overall score for the problem. The developed thinking skills make it easier to understand the nature of programming and allow learning to program productively (Mayer and others, 1986).

In order to ascertain learners' attitude to teaching programming, pilot research has been carried out (using questionnaire method) involving respondents from groups of different completed programming course: 29 respondents from information systems technology degree program, 13 - studying programming engineer, 13 - studying informatics, 12 studying mathematics and computer science specialty and 9 - studying programming and internet technologies. A total of 76 students were interviewed. Almost half of the respondents, which are 48.68% of the respondents, said that it is difficult to learn how to program, 39.47% of the respondents indicated that programming is not difficult to learn. Students who in the future profession will have programming, mostly chose an answer that it is not hard to learn how to program (88.46%), while those who will not have to deal with programming in the future, said that it is difficult to learn how to program (56%). Upon request, to identify the reasons why they think it is difficult to learn how to program, the respondents were self-critical enough, and pointed out that the difficulty stems from the fact that many learners are not sufficiently working independently (as stated 44.74% of respondents) and the fact that they lack motivation (47.37%). Complexity of the syntax of programming languages was also distinguished (30.26%); they stated that it is not difficult to write a program, but to move through the development process (36.84%). The survey identified training needs of the learners, i. e. learning when having individual consultations (70.9%); learning using technologies (49.1%) (Jadzgevičienė, Urbonienė, 2013).

In order to make learning programming more appealing and easy there is a need for both the teacher and learner to show efforts, attitudes, favorable disposition, close cooperation. It is also very important to select (choose) appropriate - effective in respect

of learning - programming and learning tools.

Educational programming language criteria were analyzed by many scientists (Kölling, Koch, Rosenberg, 1995; Milbrandt, 1993; Parker, Chao, Ottaway, Chang, 2006; Mannila, de Raadt, 2006). Their work analysis is summarized in Section 2.1.4 of the dissertation. At the same time there are lots of discussions in which programming language is the most appropriate for teaching beginners how to program (Duke and others, 2000; Gee and others, 2005; Gupta, 2004; Bishop, 1997). In the dissertation (2.1.4. section) by L. Mannila and M. de Raadt presented list of 17 criteria of the most popular programming language for teaching are analyzed *C, C++, Eiffel, Java, JavaScript, Logo, Pascal, Python, Visual Basic*.

Teaching programming must be based on the general laws of psychology and pedagogy, taking into account the specific requirements to anticipate problems. It is therefore useful to teach programming by applying Bloom's, SOLO (Bloom, 1956, Lister and others, 2003; Lister and others, 2004) and other cognitive sciences taxonomies by combining them into the curriculum. Developing APMS used programming teaching course will be based on Bloom's taxonomy and the SOLO taxonomy will not be directly used. Preparing curriculum of the teaching subject it must be taken into account that to formulate professional competencies active verbs complying with Bloom's taxonomy levels must be used, named in section 2.5.1. To describe learning course, it is necessary to know topics that compose the course and topics that correspond the desirable specific competences. The author of the work recommends relying on *Computer Science Curriculum 2008* (CS2008) and *Computer Science Curriculum 2013* (CS2013).

Analyzing the programming learning characteristics it was observed that learning programming is a complex intellectual process that requires understanding of the programming concepts in the field, knowledge of their usage, study of various examples, besides learner’s active participation in solving similar tasks is important. Considering these features, educational activities should be used and respective types of LO combined to them. Personalization of learning is also important, since the learner is characterized by a distinctive style of information processing, considering which it would be possible to increase learning efficiency. As pilot study demonstrated, learners themselves also emphasize the need for individualization of the learning process.

2.2. Analysis of the existing teaching programming tools and environments

In Lithuanian educational institutions various environments used for teaching programming are found. Schools still teach programming with *Pascal* programming language and FreePascal environment is used to realize the language. Universities teach teach programming using *C + +, Java, VisualBasic* programming languages, in some places *Phyton* was started to be used. These languages use a variety of environments: C++ – *Borland C++, Dev C++, VisualStudio, CodeBlocks*; *Java – JBuilder, Eclipse, NetBeans, and BlueJ*. Also, for the analysis, programming environments *Alice* and *Scratch* were chosen, which allows creating interactive stories, cartoons and simple video games just with the mouse. In the section 2.2. of the dissertation, analysis of the programming environments defined by 32 criteria is presented. The analyzed programming environments are not suitable for beginners to learn programming, because they do not allow providing learning materials properly (in the learner’s native language) and receiving a fast, functional and formative feedback. It is true that outside environment (in the websites and books) variety of learning materials can be found

(called tutorials). The environments have no adaptation and there are sufficiently limited customization options (Urbonienė, 2012).

Sorva (Sorva, 2012) examined various visualization tools for programs (*Basic Programming, LOPLE, DYNAMOD, DynaLab, Amethyst, Bradman, EROSI, VisMod, Fernández and others tool, DISCOVER, VINCE, ORGE, JAVAVIS, Seppälä tool, OOP-Anim, JavaMod, JIVE, Memview, JavaTool, PlanAni, Eliot, Jeliot, GRASP/jGRASP, VIP, The Teaching Machine, ViLLE, Jype, the Online Python Tutor, WinHIPE, CSmart, Online Tutoring System*) and their characteristics in his dissertation. The above mentioned examined tools allow visualizing a variety of learning activities, but none have possibility to personalize learning according to the learner's learning style.

2.3. Adaptive technologies and learning systems

The adaptive system adapts itself to the individual learner's characteristics and needs.

Shute and Zapata-Rivera (2008) define four processes of the cycle (with a variety of possible adaptation scenarios, Figure 1.), allowing the learner to achieve the relevant learning content and resources. This cycle consists of:

- *Capturing* - includes collection of personal information about the learner while he is interacting with the environment;
- *Analysis* - a process creates and maintains a certain area of learner‘s model, later presenting information taken into account the current state findings (from the accumulation phase);
- *Selection* - learning material is chosen according to learner's model supported by the system and by system objectives (for example, other LO)
- *Presentation* - regarding the results of selection process, the learning content is presented corresponding to the learner's needs and specific characteristics.

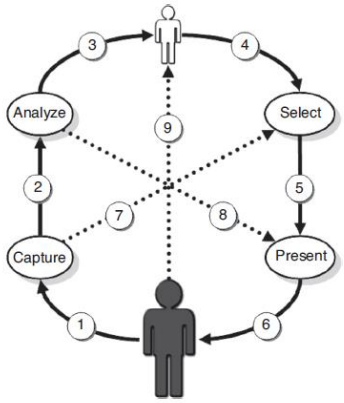

Figure 1. **Application model of the adaptive technologies using four processes** (Schute, Zapata-Rivera, 2008)

Adaptive systems are created by integrating several technologies into one system. Developing adaptive systems can be adapted different variables of the learner and teaching. It depends on the use of the adaptive system. APTS is adapted by educational material review.

Brusilovsky and Maybury (2002) define adaptive system model shown in Figure2:

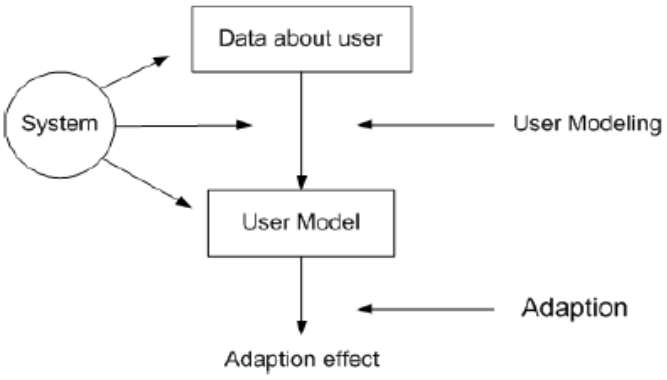

Figure 2. **Model of adaptive learning system** (Brusilovsky, Maybury, 2002)

Adaptation in e-learning is based both on the adaptation process and the state in which they are adapted. Adaptive term here is associated with a large range of system features and capabilities. Different types of adaptation are distinguished: advance or reactive, autonomous or planned, total or selective (Moura, 2006).

The learning environment is considered adaptive if it is able to (Paramythis, Loidl-Reisinger, 2004):

- monitor the activities of its users;
- interpret these actions based on specific area;
- in accordance with interpreted actions to determine the user's requirements and regulations so that they can be shown in the associated model;
- affect the allowable knowledge so that in view of the user and the topic area dynamically would facilitate the learning process.

Brusilovsky and Henze (2007) distinguish the main adaptation models according to the techniques used:

- *Manual indexing of learning resources.* The person selects elements necessary for adaptation and saves in relevant documents, later these items are used to locate optimal compliance and for assessment;
- *Automatic indexing based on keywords.* The required elements of adaptation are defined by keywords that are used to locate and assess optimally compliance;
- *The metadata-based adaptation.* Elements that are necessary for adaptation are defined by metadata that are used to find and assess the optimal compliance; precisely this method was used in designing the APTS.
- *Community-based adaptation.* When the report on optimal compliance with suitable elements for the adaptation is based on the same characteristics with other users' experiences. (Brusilovsky, Henze, 2007)

Hauger and Köck (2007) performed analysis of the adaptive learning systems *AHA, ALFANET, ELM-ART, ITEM/IP, InterBook, KnowledgeSea II, METOD, NetCoach* and others. In the above mentioned analyzed systems, help and support are adaptively provided, links are shown or hidden. In some systems, for example *KnowledgeSea II*, personalization by allowing learners to navigate and access recommended information is used, elsewhere for example, in *Knowledge Tree* adaptability is supported in terms of learning activities. Among the examined adaptive systems, there are a few used for teaching programming, for example *iWeaver* is used to teach the *Java* programming language, when after determined learner's personal characteristics personalized

recommendations are presented and respectively, accessible learning tools in the user interface are displayed. Although there are adaptive learning systems developed, in none of them learning materials are created automatically, or when providing teaching, learner's learning style is not taken into account.

Eberhart, Shi (2007) distinguishes three types of system adaptation:

- Maintained adaptation - in this case "tutor" who provides certain specific examples (specimens) participates;
- An enhanced adaptation - adaptation takes place through interaction between system and heuristic information (certain criticism), which amplifies the effect of adaptation. Precisely this method was used in designing APTS;
- Unattended adaptation - in this case, there is neither tutor nor criticism.

Analysis of the literature sources and scientific articles showed that there are no adaptive systems of programming learning, that would automatically create learning course and which could be adapted to the learner's individual characteristics depending on his learning styles.

2.4. Software agents and agent systems

Whereas while teaching programming and developing learning course and programming tools multitude of problems are encountered - there is a need to choose the programming language and paradigm, the programming environment, to define the educational specific competences and enable them to define learning topics and learning activities, to foresee and prepare learning materials, teaching and evaluative tasks (which allow to validate specific competences). Each different selection requires considerable effort from the course developer. Therefore, it is appropriate to create a system that would be universal and will automate major part of the course developer's routine work, taking into account the information regarding educational curriculum. Training course specified in educational curriculum content using auto-tagging of program and using LO as training materials can be created. It is appropriate to use software agents in order to automate factors of actions and reactions to environment in the system.

The software agent is an autonomous part of the software that performs user's instructions or programmed situation decisions. The main idea of agents is that they are not strictly summoned to perform a certain task, but are themselves active. Depending on the type, they can learn, draw conclusions, and communicate with each other to achieve common goals to change the location (Lupeikienė, 2007). Agents follow principles of beliefs, desires and intentions *(BDI – Belief, Desire, Intention).* According to this principle, when the result of some event occurs, the agent responds to it on the basis of his knowledge (*beliefs*) and goals they want to achieve in this event (*desire*). Objectives are implemented according to predefined plans (*intentions*). Agents are modeled as having their own goals, actions and knowledge. Typically, several agents are used in the systems. While acting, they interact with each other, so forming multiagent system. APMS should perform a different kind of actions, so it will be modeled as a multiagent system where each specific action will be performed by a separate agent (see Chapter 3). Reactive agents will function in the system.

2.5. Learning styles

Learning activities, usable learning methods, raised questions, communication and generalization depends from learner's style. There is also a very important part of mental activity, which leads to algorithmic and logical thinking. *Herrmann* learning style classification is based on the human brain's hemispheres work. Based on this model, it's possible to identify expression of the degree of one of the four individual quadrants (A, B, C and D) (Fig. 3). This is determined by using a special questionnaire of 120 questions (*HBDI - Herrman Brain Dominance Instrument*) and analyzing the results.

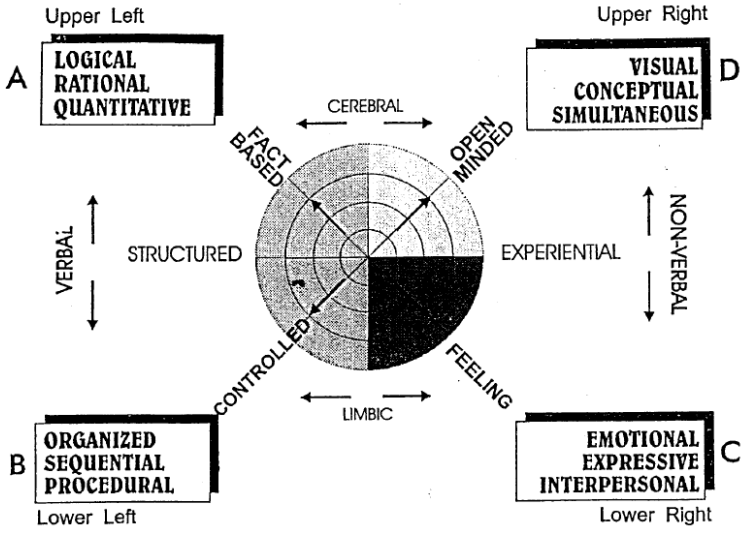

Figure 3 **Whole brain teaching and learning model** (by Herrmann)

This method allows learners to be divided according to their way of thinking - information processing - nature. According to the physical functioning principles of the brain during the tasks 4 states (quadrants) are allocated. However, it may well be that expressed enough are features of the two corresponding (this is typically 58% of the population) quadrants, or there can be characterized expression level by the equality of all four quadrants (there are about 3% of such people) (Lumsdaine, Lumsdaine, 1995). Knowing learner's individual options it is possible to search for individual learning activities, tools and learning content elements (LO), or allow him to choose the most suitable for him.

Analysis of the examined sources indicated that learning object types suitable for programming learning were not distinguished and described they were not related to the classification of learning styles by Herrmann. Information and analysis provided in the research works, makes a fundamental distinction between programming learning activities: study of the sources of information (concepts, terminology, concepts); study of samples; study of graphic information; study of visual material; modeling and simulation; tasksolving; problem-solving; drafting. These activities influence selection of the LO types used in the learning course.

In addition, study of the scientific works showed that there was no mention of adaptation of teaching subject items (LO) according to the learner's learning style by Herrmann's classification. Since the projected learning system aims to enhance learning efficiency, which is determined by the information used in learning process and processing by individual human potential, as well as finding the required information

fast, with APTS it is necessary to take into account the individual learner's characteristics and to adapt learning materials in learning process according to that.

2.6. Elements of learning material and their repositories

So far, there is no universally accepted formulated dictionary of the educational activities and teaching methods, while LO is a universally accepted standard so APMS learning material is associated with LO. The learning object (LO) - is any digital resource that can be used for learning and newly applied in other learning contexts (Coughlin, Kurilov, 2008, Wiley, 2000). LO is not only the object (image, text, and so on), but also material describing metadata related to that, where object's author, title, purpose, language, subject, type, etc, are referred to. According to this data object can be found in LO repositories. Metadata standards (*IEEE LOM, Dublin Core, ADL SCORM, ISO Metadata Standard MLR* and so on.) and specifications (*IMS Learning Resource Meta-data Specification*, etc.) used in LO repository allows to describe the elements of adaptation unanimously - learning resources, learners, activity architecture and content packages. In Europe international *LOM* standard and its application models are widely used. *LOM* (*Learning Object Metadata*) – is data model, usually encoded in *XML*, and is used to describe learning objects and related digital resources designed to support learning. By this standard the design of APMS is based.

In order to achieve reusability of system components, subject material can be composed of LO that are stored in LO repositories. Learning quality should be assured evaluating the quality of LO prior to their use in training. In order to select and use the appropriate LO, stored in different repositories LO retrieval methods are used. One of such methods is *Harvesting*. For data retrieval independent client data retrieval program is used based on metadata (*Harverster*), which carries *OAI-PMH* queries (*Open Archives Initiative Protocol for Metadata Harvesting*).

Elected LO from LO repositories needs to be adapted to one's created teaching subject's course. As mentioned in section 2.1.6., APTS training course elements (LO) will be associated with the targeted subject topics, professional competences and *Bloom's* Taxonomy levels. This linking can be made by a person - learning course developer. However, the course development process can be automated. The analysis failed to detect precise compliance of the curriculum (topics and subject-specific competences) with LO and *Bloom's* taxonomy levels. Since in projected Adaptive programming learning system professional competencies' evaluation is provided with regard to *Bloom's* taxonomy levels, the above mentioned compliance should be implemented. To mark the APTS course, the extended partitioning method proposed by Kurilov (2008) is used. LO will be marked (see 3.4. Chapter) by recording information about the suitability for the topic, subject competence and *Bloom's* taxonomy level.

2.7. Training systems development models

To develop adaptive programming training model applicable ADDIE training development model (Figure 4)is applied, the stages of which can be specified as follows: analysis - preparation of the analysis of learners characteristics and learning objects, design - adaptation of programming teaching subject description, development - adaptation of programming learning material that is marking of learning objects by

curriculum description, implementation - adaptive presentation of subject course to the learner.

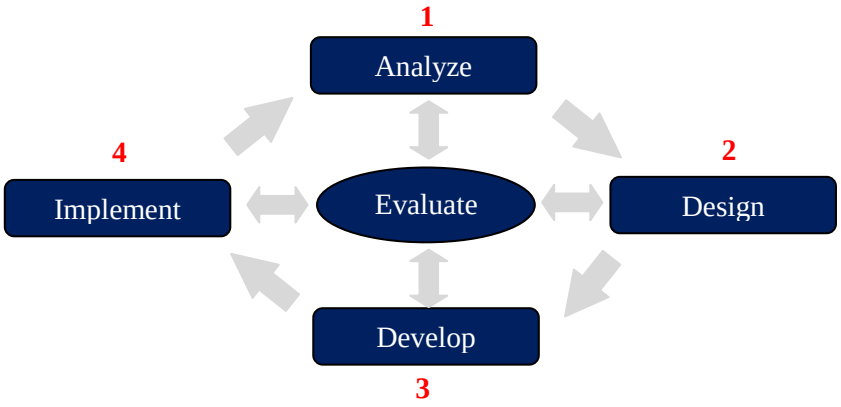

Figure 4 **ADDIE model scheme** (by Davis, 2013)

To model the system the conceptual model is chosen - the visual method (diagram) representing the causal relationships among the factors which are believed to be relevant to the analyzed issue or subject. A good model should clearly link object to the direct-acting factors (direct threats or opportunities), circumstantial factors and strategic activities that influence those factors. The conceptual model is accompanied by a text description which explains the made up diagram in words. Creating a conceptual model all the elements that are included in the model and the rules defining behavior of those elements are described.

3. DESIGNING OF ADAPTIVE PROGRAMMING TEACHING SYSTEM

Scientific studies showed that there are no such systems, which would allow automatically create and adapt programming training course (PTC, which is educational curriculum and its components (LO)) to learner's individual characteristics according to his learning style. Based on Chapter 2 conclusions, it can be said that it is appropriate to develop an adaptive system based on LO-compliance with *Herrmann's* learning styles and MO-compliance with the curriculum (topics, professional competencies) and *Bloom's* Taxonomy levels. By the system's basis can be chosen specialized repository of programming LO metadata and tags (user-created metadata) (SLOR), enriched with intelectual technologies - multiagent system. This multiagent systems foundation must have program agents which implement the above-mentioned characteristics of the system - Herrmann's learning styles compliance with programming learning appropriate LO types and curriculum compliance with programming LO and *Bloom's* taxonomy levels that define achievement of professional competencies. In projected system enhanced adaptation method (see Chapter 2.4.) is used, when the adaptation is performed on the basis of expert assessment results when ranking types of the LO. To assure system adaptivity and quality additional agents are used as well:

- quality LO metadata of programming subject area's retrieval from the indicated LO repositories to the SLOR,
- software agent of LO types ranking,
- adaptive PTC provision;

- PTC subject competency assessment.

So APTS consists of SLOR along with the software agents that help with adaptation functions (Fig. 5). SLOR is created by MySQL database basis. The projected APTS will use MO, which are held in various already existing previously created MO repositories. Adaptation will be based on metadata (see chapter 2.4.). In order that it would be used only high quality and educational subject context corresponding LO, their metadata information is stored primarily in a specialized SLOR. These LO defined by adaptation rules will be used by learners with individual characteristics. Learner's in adaptive systems model provides learner's image and applies training material and activities according to learner's individual characteristics and previous learning experience (chapter 2.4.), but APTS will only be used in learner's profile stored information about his unique style of learning (by *Herrmann's* classification).

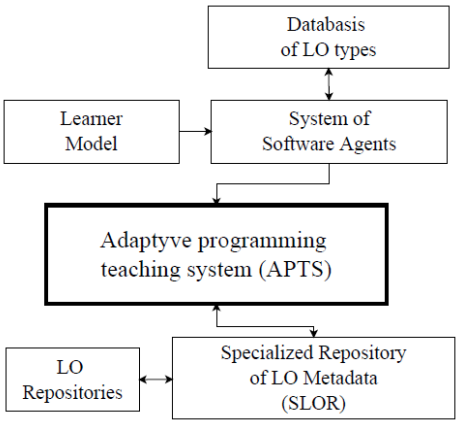

Figure 5. **Model of APTS context**

APTS has been designed as a multiagent system (Figure 6), in which 6 program agents is running that allows to create adaptive PTC, which adapts elements of the curriculum (LO) according to the learner's learning style by Herrmann.

LO metadata and tags are stored in SLOR. Learner's learning style is determined by a 120-question test and the test results are stored in learner’s profile.

So agents operate APTS:

1. *LO types ranking agent* - by expert evaluation method selected LO types for programming teaching are ranked determining eligibility to learner's learning style according to percentage values of Herrmann's learning style classification.
2. *LO election agent* - using Harvesting method (additional agent) programming curriculum corresponding LO are selected from LO repositories, which allows to store tags of users star rating and the number of downloads. Before downloading LO to SLOR is evaluated LO quality - are compared users star rating (must be > 2.5) and the download number (must be > 3) tags. High-quality programming MO metadata are stored in SLOR.
3. *PTC development agent* - analyzing provided programming learning program is performed programming LO linking to the offered topics in the program, professional competencies and *Bloom's* Taxonomy levels.
4. *Adaptive PTC delivery agent* - helps the learner to learn purposeful channeling to repeat unsuccessfully mastered topics - if the subject task is performed without the adequate minimum defined level, the student is returned to repeat the topic - a list of MO is provided to him, where most relevant LO presenting learner's learning style

are provided first. Initially, the learner has access just to the content and task type LO (LO_C and LO_T). Evaluation LO (MO_E) becomes available only when the learner performs all the tasks (uses all LO_T) at least at a minimum level of defined tasks and the overall assessment is > 45%. Providing course for the learner additionally operates *learning style monitoring agent*, which recalculates the percentages of the types of LO according to the learner's match to Herrmann's learning style expression of values, and depending from the obtained values accordingly provides the learner with LO.

5. *Competency evaluation agent* - allows evaluating achievement of professional competencies. The learner gets access to the assessment type LO. After the evaluative task are done (using assessment type LO - LO_E) competency level is determined - the highest level of competence is when learning outcomes (LR) will be 85-100% (or respectively 9-10 in ten-point grading system), the average level of competence - when LR will be 65-84% (or 7-8), mandatory minimum competency level - when LR will be 45-64% (or 5-6).

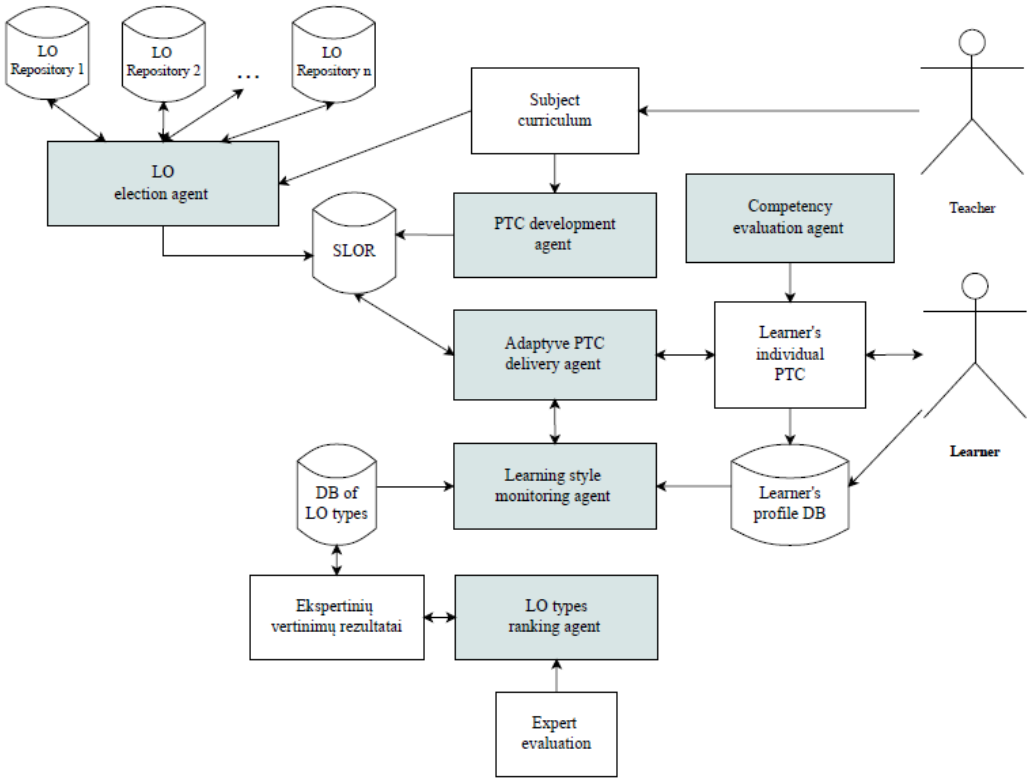

Figure 6. **Scheme of the APTS model**

In figure 7 sequence of in APTS acting agents' interconnection and operation is provided.

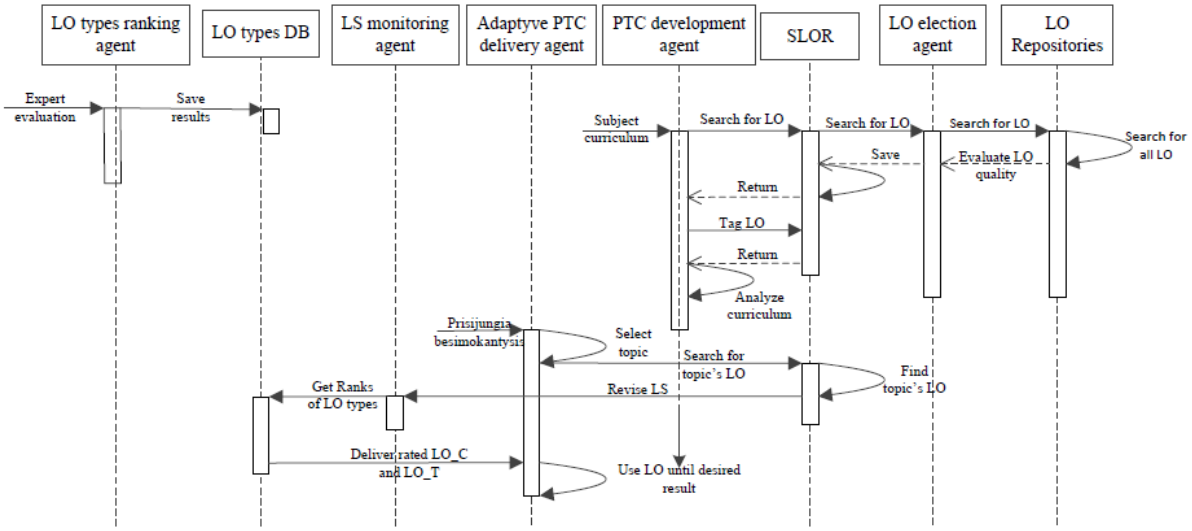

Figure 7. O**peration sequence diagram of the APTS agents**

In the dissertation, APTS conceptual model is designed, that uses intelligent technologies - software agents and conceptual model that helps to automate the teaching subject course's development process and to adapt learning materials to learner taking into account his *Herrmann's* learning style.

To create the model the expert quality evaluation of learning object types which allows ensuring the quality of learning course material was used. A subject course's automated tagging enables to refuse human factors which influence the course development that determines the greater likelihood of errors.

4. Evaluation of the model

To assess the quality of the APTS model in respect of usage, expert assessment method is selected.

Experts for the evaluation of the generated APTS model have to meet certain requirements. The expert must have:

- doctoral degree in Informatics or Informatics Engineering;
- at least 5 years' experience of teaching programming;
- understanding of the LO use for learning;
- at least one research paper in reviewed research journals printed.

To estimate APTS model, expert evaluation method of the results processing e.g. performed ranking (rating) task procedure is used. For that, a set of system evaluation criteria is defined, and rated assessment values established. To evaluate the system's model APTS models evaluation criteria (Table 1) (based on *ISO/IEC FCD 25010*, 2009) was chosen.

Table 1. **APTS models evaluation criteria** (adapted by *ISO/IEC FCD 25010*)

No.	Characteristics	Sub characteristics	Description	Evaluation scale
1.	Efficiency	Efficiency	Suitable LO are provided to the learner which allow to pursue professional competencies provided in the subject curriculum	5 – very big 4 – big 3 – average 2 – small 1 – very small

No.	Characteristics	Sub characteristics	Description	Evaluation scale
2.	Capacity/ Effectiveness	Capacity	Methods proposed in the model allow to save time in choosing the appropriate LO suitable to perform specific tasks	5 – very big 4 – big 3 – average 2 – small 1 – very small
3.	Productivity	Productivity	In order to reach efficiency in model, proposed methods allow the user to increase the amount of LO	5 – very big 4 – big 3 – average 2 – small 1 – very small
4.	Usability	Flexibility	In the model proposed methods allow to adapt the subject material to the learner's individual characteristics	5 – very big 4 – big 3 – average 2 – small 1 – very small
		Context conformity	In the model proposed methods can be used in the provided learning contexts	5 – very big 4 – big 3 – average 2 – small 1 – very small

Defined APTS model's usage quality, expert evaluation criteria, rating scale value markings are provided in Table 2.

Table 2. **APTS model‘s assessment criteria meaning**

Value	Values of evaluation criteria	Marking
5	Very big	VB
4	Big	B
3	Average	A
2	Small	S
1	Very small	VS

After defining the evaluation criteria, interview with APTS evaluation model’s experts was conducted. During the interview the designed APTS model was presented and experts were asked to choose the provided values of evaluation criteria. Experts’ evaluation results are presented in Table 3.

Table 3. **APTS experts’ evaluation results**

No.	Criteria	Experts evaluation						
		1	2	3	4	5	6	7
1.	Efficiency	VB	VB	B	VB	B	B	VB
2.	Capacity	VB	VB	VB	VB	B	VB	VB
3.	Productivity	VB	VB	B	B	A	B	B
4.	Flexibility	VB	VB	VB	VB	B	VB	B
5.	Context conformity	VB	VB	VB	VB	B	VB	B

After applying multiple criteria decision-making technique (Skūpienė, 2010) and after using *Fuzzy* numbers lingvistic values (Table 4), expert evaluation turned into a numerical value, which specifies the expression of criterion (Table 5). Other calculations use middle values of triangular *Fuzzy* numbers: VB – 1.0, B – 0.8, A – 0.6, S – 0.4, VS – 0.2.

Table 4. **APTS experts' evaluation results**

Lingvistic term	Tiangular Fuzzy numbers value
Very big / very high	(0.8; 1.0; 1.0)
Big / high	(0.6; 0.8; 1.0)
Average	(0.4; 0.6; 0.8)
Small / low	(0.2; 0.4; 0.6)
Very small /very low	(0.0; 0.2; 0.2)

Table 5. **Converted experts' evaluation results into triangle indistinct numbers**

No.	Criteria	Weight	Values of experts evaluation							Average (a_j)
			1	2	3	4	5	6	7	
1.	Efficiency	0.2	1.0	1.0	0.8	1.0	0.8	0.8	1.0	0.9143
2.	Capacity	0.2	1.0	1.0	1.0	1.0	0.8	1.0	1.0	0.9714
3.	Productivity	0.2	1.0	1.0	0.8	0.8	0.6	0.8	0.8	0.8286
4.	Flexibility	0.2	1.0	1.0	1.0	1.0	0.8	1.0	0.8	0.9429
5.	Context conformity	0.2	1.0	1.0	1.0	1.0	0.8	1.0	0.8	0.9429

All decision-makers is equally important, their estimates x_{ij}, where $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$, have the same weight d_t=0,14, where t>2 ir $\sum_{t=1}^{n} d_t = 1$. Arithmetic means of all criteria a_j are calculated (Table 5).

Set of APTS model evaluation criteria as well of equal importance, so their wigts s_j, where $j = 1, 2, ..., m$, are also the same and they are equal 0,2 ir $\sum_{j=1}^{m} s_j = 1$.

Overall evaluation of the quality of the conceptual model, which is defined by the function f (x) (formula 3), is calculated:

$$f(x) = \sum_{j=1}^{m} w_j \cdot a_j \qquad (3)$$

Applying the formula the result is:

$$f(x) = 0.2 \times 0.9143 + 0.2 \times 0.9714 + 0.2 \times 0.8286 + 0.2 \times 0.9429 +$$

$$+0.2 \times 0.9429 = 0.92$$

Expert evaluation result (0.92), depending on triangular fuzzy numbers conversion to linguistic variables values (Table 4), is transformed into a linguistic variable and is determined that the overall evaluation of the conceptual model is very high, that is corresponds to 92 % of absolute quality (which is equal to 100%).

The results also show that the APTS model efficiency is very high (91.4%), productivity - very high (97.1%), productivity - high (82.9%), flexibility and context conformity - very high (94.3%).

5. GENERAL CONCLUSIONS AND RECOMMENDATIONS

1. After analyzing programming teaching problems, the existing programming and adaptive teaching systems, influence of learners' learning styles on learning outcomes, the following was determined:
   1.1. Personalization of learning is important because a learner is characterized by a distinctive style of information processing, which has to be considered in seeking to increase learning efficiency. As pilot study revealed, the need for individualization of learning is also emphasized by learners.
   1.2. There are no adaptive programming learning systems that automatically would create learning course and it would be adapted to learner's individual characteristics considering his learning style. Since teaching to program and in preparing training course there are multitude of problems, it is appropriate to create a system that would be universal and would automate major part of the course developer's routine work, taking into account information contained in the educational curriculum. Using LO and programs automatic tagging, learning course corresponds content indicated in the curriculum can be developed. It is appropriate to use software agents for automatization of actions and reactions to environmental factors in the system.
   1.3. Learning object types suitable for programming learning and connected to learning styles by *Herrmann's* learning styles classification were not singled out and described. This association provides individualization of learning, when taking into consideration the individual learner's characteristics; learning materials are adapted when learning. The selection of LO types used in learning course is affected by learning activities which are identified in the time of analysis: study of the information sources (conception, terminology, concepts); the study of samples; the study of graphic information; the study of video material; modeling and simulation; task solving; problem-solving; drafting.
   1.1. Precise linking of the curriculum (topics and subject-specific competencies) with LO and *Bloom's* taxonomy levels should be performed, to ensure evaluation of professional competencies considering *Bloom's* taxonomy levels in the projected adaptive programming learning system.
2. Based on the programming teaching features, learner's individual characteristics and by methods of adaptive learning systems which were determined during the analysis, adaptive programming learning system's model was created, which:
   2.1. Links *Herrmann's* learning styles with LO types. The developed expert evaluation method of LO types allows better LO adaptation to learner's learning style;
   2.2. Provides opportunity to evaluate quality of the LO before offering them to learning course. The integration of the LO qualitative evaluation method to the APMS ensures quality of learning material because in specialized learning objects metadata repository (SLOR) only qualitative LO are stored;
   2.3. Allows linking LO with the learning course topics, professional competencies and *Bloom's* Taxonomy levels and allows assess the achievement of the professional competencies according to pre-defined rules;
   2.4. Allows to personalize learning, that means, to recommend to the learner the course which is most suitable for his individual characteristics (according to

learner's learning style), while choosing a number of LO types which match his learning style,

3. The expert evaluation of adaptive programming teaching system conceptual model showed that:
   - The model proposed methods allow to provide the learner appropriate LO suited to pursue professional competencies provided in the subject curriculum, and it can be said that the system is effective;
   - The model proposed methods allow to save time in choosing the specific tasks appropriate LO, so the system is efficient;
   - The model proposed methods allow the user to increase the amount of LO, so methods are productive;
   - The model proposed methods allow to adapt the subject matterial to the learner's individual characteristics, the system is flexible;
   - The model proposed methods can be used in different learning contexts, thus ensuring context conformity.
4. Using the proposed model can be created adaptive programming teaching system which adapts automatically generated subject matterial to the learner and be characterized as effective, efficient, productive, usabile (flexibile and context conformable).
5. Using the proposed model of the system, it is possible to create a system that offers the following parameters: effectiveness, efficiency, productivity and usability (flexibility and context conformity). This system can be integrated as separate module in to existing virtual learning environments like Moodle.
6. The proposed model can be applied to other areas of learning, respectively distinguishing types of learning object the most suitable for learning and in a curriculum listing naming the themes, subject-specific competences and Bloom's taxonomy levels.

LIST OF LITERATURE REFERENCES IN THIS SUMMARY

1. ACM/IEEE-CS, 2008. Computer Science Curriculum 2008: An Interim Revision of CS 2001, ACM/IEEE-CS Joint Interim Review Task Force Report, ACM Press.
2. ACM/IEEE-CS, 2012. Strawman Draft: Computer Science Curricula 2013.The Joint Task Force on Computing Curricula Association for Computing Machinery and IEEE-Computer Society.
3. Baziukaitė D., Vaira Ž., Idzelytė D. (2008). A tool to support self-education in a lifelong learning, Innovative techniques in instruction technology, e-learning, e-assessment and education. New York: Springer, 2008, p. 92-97.
4. Bennedsen J. (2008). Teaching and Learning Introductory Programming – A Model-Based Approach. *Dissertation for Dr. Philos degree in the Faculty of mathematics and Natural Sciences,* University of Oslo, Norway, 2008
5. Bishop J. M. (1997). A philosophy of teaching Java as a first teaching language, *in Proceedings of SACLA 1997 Conference*. [interactive]. [previewed 22/03/2012]. Available at: http://www.cs.up.ac.za/cs/jbishop/Homepage/Pubs/Tech-reports/SACLA97.pdf

6. Blonskis J., Dagienė V. (2003). Programavimo pagrindų mokymo vidurinėje ir aukštojoje mokyklose lyginamoji analizė. *Informacijos mokslai*, ISSN 1392-0561, 26, 2003, p. 23–28
7. Bloom B. S. (ed.) (1956). *Taxonomy of Educational Objectives, the classification of educational goals: Handbook I - Cognitive Domain*, New York: McKay
8. Brusilovsky P., Henze N. (2007). Open Corpus Adaptive Educational Hypermedia. In P. Brusilovsky, A. Kobsa & W. Nejdl (Eds.), *The Adaptive Web*: Springer Berlin Heidelberg. Vol. 4321, p. 671-696
9. Brusilovsky P., Maybury M. T. (2002). From adaptive hypermedia to the adaptive web. *Communications of the ACM*, vol. 45, no. 5, 2002, p. 30–33.
10. Caspersen M E. (2007). Educating Novices in The Skills of Programming. *PhD Dissertation*, Department of Computer Science University of Aarhus, Denmark, February 2007.
11. Dagienė V., Kurilovas E. (2008). *Informacinės technologijos švietime: patirtis ir analizė: Monografija*. Vilnius: Matematikos ir informatikos institutas. 216 p.
12. Davis L. A. (2013). Using instructional design principles to develop effective information literacy instruction. The ADDIE model. [interactive] [previewed 19/12/2013] Available at: http://crln.acrl.org/content/74/4/205.full
13. Duke R., Salzman E., Burmeister J., Poon J., Murray L. (2000) Teaching Programming to Beginners Choosing the Language is just the First Step, ACE 2000, Melbourne, Australia, *ACM proceedings*, p. 79-86.
14. Ebel G., Ben-Ari M. (2006). Affective Effects of Program Visualization. *ICER '06 Proceedings of the second international workshop on Computing education research*. Canterbury, UK, 2006, p. 1–5. ISBN:1-59593-494-4
15. Eberhart R. C., Shi Y. (2007). *Computational Intelligence. Concepts to Implementations*. Morgan Kaufmann Publishers, United States of America. ISBN 978-1-55860-759-0
16. Franzoni A. L., Assar S. (2009). Student Learning Styles Adaptation Method Based on Teaching Strategies and Electronic Media. Educational Technology & Society, 12 (4), p. 15–29.
17. Gee Q., Wills G., Cooke E. (2005) A first programming language for IT students. In, *6th Annual Conference of the LTSN Centre for ICS, York, UK, 30 Aug - 01 Sep 2005*
18. Gupta D. (2004) What is a good first programming language? *ACM proceedings*, Volume 10 Issue 4, August 2004
19. Hauger D., Köck M. (2007). State of the Art of Adaptivity in E-Learning Platforms. In: Brunkhorst, I., Krause, D. & Sitou, W. (Hrsg.), *15th Workshop on Adaptivity and User Modeling in Interactive Systems*.
20. Hawk T. F., Shah A. J. (2007). Using Learning Style Instruments to Enhance Student Learning. *Decision Sciences Journal of Innovative Education*, Volume 5 Number 1, January 2007 U.S.A., p. 1-19.
21. Jadzgevičienė V., Urbonienė J. (2013). The Possibilities of Virtual Learning Environment Tool Usability for Programming Training. *Proccedings of the 6th International Conference Innovative Information Technologies for Science, Business and Education*, IIT-2013 November 14-16

22. Kannangara D. N. P. (2013). Teaching Tools and Techniques for Efficient Teaching and Learning of Computer Programming for Beginners Using JAVA. *Thesis for the Degree of Doctor of Philosophy of Curtin University*
23. Kinnunen P. (2009). Challenges of teaching and studying programming at a university of technology – Viewpoints of students, teachers and the university. *Dissertation for the degree of Doctor of Philosophy of the Faculty of Information and Natural Sciences at Helsinki University of Technology* (Espoo, Finland), December.
24. Kölling M., Koch B., Rosenberg J. (1995). Requirements For A First Year Object-Oriented Teaching Language**.** *Proceedings of 26th SIGCSE Technical Symposium on Computer Science Education*, 1995, Nashville, Tennessee, U.S.A., SIGCSE Bulletin 27, 1, March, p. 173-177.
25. Kurilovas E. (2008). Digital Library of Educational Resources and Services Components Interoperability Problems. *Doctoral Dissertation*, Vilnius Gediminas Technical University, Institute of Mathematics and Informatics, Technological Sciences, Informatics Engineering (07 T).
26. Kurilovas E. (2012). European Learning Resource Exchange – a platform for collaboration of researchers, policy makers, practitioners, and publishers to share digital learning resources and new e-learning practices. *Chapter 14 in the book: Ahmed Cakir and Patricia Ordóñez de Pablos (Ed.) „Social Development and High Technology Industries: Strategies and Applications"*. IGI Publishing, USA, 2012, p. 200–243, ISBN 978-1-61350-192-4
27. Lister R., Leaney J. (2003). First Year Programming: Let All the Flowers Bloom. *5th Australasian Computer Education Conference (ACE2003)*, Adelaide, 2003, Vol. 20.
28. Lister R., Simon B., Thompson E., Whalley J. L., Prasad Ch. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *SIGCSE Bull.*, 2006, 38(3), p. 118–122.
29. Lumsdaine M., Lumsdaine E. (1995) Thinking Preferences of Engineering Students: Implications for Curriculum Restructuring. *Journal of Engineering Education*. April, 1995, Vol. 84, No. 2. [interactive] [previewed 15/12/2013]. Available at: http://www.innovationtoday.biz/pubs/Thinking_Preferences_of_Engineering_Students.pdf
30. Lupeikienė A. Teoriniai ir technologiniai informacinių sistemų aspektai. Išleista Matematikos ir informatikos institute. UAB „Mokslo aidai", 2007. ISBN 978-9986-680-40-6
31. Mannila L. (2007). Novices' Progress in Introductory Programming Courses. *Informatics in Education*, 2007, Vol. 6, No. 1, p. 139–152.
32. Mannila L., De Raadt M. (2006). An Objective Comparison of Languages for Teaching Introductory Programming. *Proceedings*, Koli Calling 2006, p. 32-37
33. Mayer R. E.; Dyck J. L., Vilberg W. (1986) Learning to Program and Learning to Think: What's the Connection? *Communications of the ACM*, July 1986, Vol. 29, No. 7, pp. 605-610
34. Milbrandt G. (1993). Using Problem Solving to Teach a Programming Language in Computer Studies. *Journal of Computer Science Education*, 1993.

35. Moura H. (2006). Adaptive e-Learning Environment Design. *Interactive Educational Multimedia*, Number 12, April 2006, p. 62–71.
36. Paramythis A., Loidl-Reisinger S. (2004). Adaptive learning environments and e-learning standards. *Electronic Journal on e-Learning* Volume 2 Issue 1, February 2004, p. 181–194.
37. Parker K. R., Chao J. T., Ottaway T. A., Chang J. (2006). A Formal Language Selection Process for Introductory Programming Courses. *Journal of Information Technology Education,* 2006, Volume 5, p. 133-151.
38. Preidys S., Žilinskienė I. (2012). Nuotolinio mokymosi kurso personalizavimo modelis mokymosi veiklų atžvilgiu. *Electronic Learning, Information and Communication: Theory and Practice (Elektroninis mokymasis, Informacija ir komunikacija: Teorija ir praktika)*. Vilnius: Vilniaus universitetas, 2012, ISSN: 23352493, p. 111-132.
39. Schute J. V., Zapata-Rivera D. (2008) Adaptive Technologies. *Handbook of Research on Educational Communications and Technology*. Taylor & Francis Group, New York, p. 277-294
40. Urbonienė J. (2012). Programming environment analysis considering learning efficiency (Programavimo mokymo aplinkų efektyvumo mokymo(si) atžvilgiu analizė). *Studijos šiuolaikinėje visuomenėje. Mokslo darbai*. ISSN 2029-431X, Nr. 3 (1) p. 243-254
41. Weber G., Brusilovsky P. (2001). ELM-ART: An adaptive versatile system for Web-based instruction. *International Journal of Artificial Intelligence in Education*, 12, 2001, p. 351-384
42. Wiley D. A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. *In David A. Wiley* (Ed.)

LIST OF PUBLICATIONS BY THE AUTHOR ON THE SUBJECT OF DISSERTATION

Articles in peer-reviewed periodical journals:

- Urbonienė J. Adaptive technologies at the study process (Adaptyvios technologijos mokymosi procese). *Mokslinės-praktinės konferencijos „Informacinės technologijos 2010: teorija, praktika, inovacijos“ pranešimų medžiaga*. ISBN 978-609-8020-10-6. Alytaus kolegijos leidykla, 2010, p. 86-91
- Dagienė V., Urbonienė J. Learning programming: comparative analysis of languages and environments (Programavimo mokymasis: lyginamoji kalbos ir aplinkos analizė). *Informacijos mokslai*. ISSN 1392-0561, VšĮ Vilniaus universiteto leidykla, 2010 54, p. 44-62
- Urbonienė J., Juškevičienė A. Research on Web 2.0 tools for programing teaching (Web 2.0 technologijų adaptuojamumo programavimo mokymui galimybių tyrimas). *Studijos šiuolaikinėje visuomenėje. Recenzuojamų straipsnių rinkinys* (įtrauktas į EBSCO). ISSN 2029-431X, Šiaurės Lietuvos kolegijos leidykla, 2011, p. 161-170
- Urbonienė J. Programming environment analysis considering learning efficiency (Programavimo mokymo aplinkų efektyvumo mokymo(si) atžvilgiu analizė). *Studijos šiuolaikinėje visuomenėje. Mokslo darbai*. ISSN 2029-431X, 2012 Nr. 3 (1) p. 243-254

- Urbonienė J. Model of an adaptive programming learning system (Adaptyvios programavimo mokymosi sistemos modelis). *Informacijos mokslai*. ISSN 1392-0561, VšĮ Vilniaus universiteto leidykla, 2013 66, p. 108-122
- Jadzgevičienė V., Urbonienė J. The Possibilities of Virtual Learning Environment Tool Usability for Programming Training. *Proccedings of the 6th International Conference Innovative Information Technologies for Science, Business and Education*, IIT-2013 November 14-16, 2013
- Jadzgevičienė V., Urbonienė J. Cooperation in Programming Learning. *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, ACM ISBN: 978-1-4503-1795-5, 2012, p. 143-144

SHORT DESCRIPTION ABOUT THE AUTHOR

Jūratė Urbonienė was born in 1974, July 30, in Siauliai city.
In 1992 she graduated from Svencionys Zigmas Zemaitis high school. In 1996 she gained Bachelor of Science in Mathematics and middle school's math and science teacher's qualification from Vilnius Pedagogical University (now Lithuanian University of Educational sciences), Faculty of Mathematics. In 1999 she gained Master's degree of mathematic science (informatics) and qualification of gymnasium teacher at the same university. In 2007 she gained Master's degree in Informatics Engineering in Kaunas University of Technology.
In 2009-2013 she studied at Vilnius University, Institute of Mathematics and Informatics, PhD (Technological Science, Informatics Engineering).
In 2010, 2011 and 2012 she was accepted into the doctoral summer school and received funding.
Since 1998 Jurate Urbonienė works in Utena College (prior to 2000 former Utena Business School) as a lecturer, and since 2009 she was appointed to supervise the Technology Department (now Electric and Informatics Engineering Department), in 2011 she was certified for associate professor's position.
E-mail: jurate99@gmail.com

SANTRAUKA

1.1. DARBO AKTUALUMAS

Programavimo mokymas yra pakankamai opi problema ne tik Lietuvoje, bet ir daugelyje užsienio šalių. Šia tema yra parašyta nemažai mokslinių straipsnių, parengta ir apginta ne viena disertacija (Bennedsen, 2008; Caspersen, 2007; Kinnunen 2009, Sorva, 2012 ir kt.), yra kelios stiprios mokslininkų grupės, kurios užsiima programavimo, algoritmavimo mokymo klausimais (pavyzdžiui, Aalto, Joensuu, Miuncheno technikos universitetuose). Šiuo metu pasaulyje aktyviai tyrinėjamas suasmenintas mokymas (-is) (toliau naudojama mokymasis), o jis gali būti siejamas su adaptyviosiomis mokymo priemonėmis-sistemomis.

Išmokti programuoti nėra lengva. Besimokantysis pirmiausia turi išsiugdyti algoritminį mąstymą. Pagrindinis šio mąstymo požymis – gebėjimas užduotį išreikšti algoritmu taip, kad jį galėtų išspręsti kompiuteris (t. y. mokėti suprogramuoti). Kokiomis priemonėmis ar metodais yra mokoma algoritminio mąstymo? Dažniausiai dėstytojas ar mokytojas supažindina su programavimo kalbos sintakse, algoritmus bei programų pavyzdžius užrašo lentoje ar parodo projektoriumi. Tai užima daug laiko ir neveiksminga. Retai naudojamos interaktyvios lentos ar projektoriai, kuriais mokytojo parašytą programą aiškiai mato visi besimokantieji ir visi kartu gali ją nagrinėti. Teigiamą poveikį mokymuisi turi informacijos vizualizavimas (Ebel, Ben-Ari, 2006), tačiau jis dažniausiai yra naudojamas kaip mokytojo pagalbinė priemonė – tiesiog mokytojas medžiagą pateikia kitokia forma, panaudodamas vaizdo medžiagą.

Akcentuojama, kad veikiant (atliekant užduotis praktiškai) žmogaus atmintis optimizuoja mokymąsi (Caspersen, 2007). Taip pat svarbu yra tai, jog dėmesio kokybė koreliuoja su mokymosi veiksmingumu – kuo daugiau laiko išlaikomas besimokančiojo dėmesys, tuo geresni jo pasiekimai (Ebel, Ben-Ari, 2006). Šiuolaikinės technologijos, kompiuteriai ir internetas suteikia galimybes mokymąsi padaryti įdomesnį ir efektyvesnį, palengvinti mokančiojo darbą, motyvuoti besimokantįjį.

Mokant ir mokantis programavimo (algoritmavimo) svarbu ne tik klaidų ar teisingų sprendimų skaičius, bet ir besimokančiojo bei jį mokančiojo asmens sąveika, komunikavimas. Dažnai ta sąveika vyksta neakivaizdžiai, per tarpininką – kompiuterį. Mokantis daug laiko sugaištama ieškant, kaip ištaisyti padarytas klaidas, kaip iš jų pasimokyti, kaip laiku gauti atsakymus į rūpimus klausimus. Tam kuriamos įvairios automatinės programavimo ir algoritmavimo mokymo sistemos (pvz. TRAKLA2, VILLE ir kt.). Šios sistemos moko programavimo ir algoritmavimo principų, tačiau jos nėra pakankamai draugiškos besimokančiajam, o ypač joms trūksta adaptyvumo.

Jau kelis dešimtmečius ieškoma efektyvesnių mokymo bei mokymosi priemonių, metodų bei būdų. Vis labiau akcentuojama individualizuoto mokymosi svarba – kai atsižvelgiant į individualias besimokančiojo savybes parenkami mokomieji objektai (MO), mokymosi veiklos, mokymosi scenarijai ir pan. Užtikrinti individualizavimą padeda ir adaptyviosios mokymosi sistemos, kurios leidžia mokymosi metu priderinti besimokančiajam tinkamiausią mokymosi turinį. Tuo pačiu bandoma pritaikyti ir pasinaudoti anksčiau sukurtais kokybiškais ir saugyklose saugomais mokymosi elementais – MO.

Mokymosi adaptavimas individualiems besimokančiojo poreikiams ir Lietuvoje, ir užsienyje analizuojamas jau senokai. Yra atlikta nemažai tyrimų, kurių metu įvairiais

metodais – atsižvelgiant į besimokančiojo savybes ir poreikius (Weber, Brusilovsky, 2001; Baziukaitė ir kt., 2008), analizuojant besimokančiojo veiklas elektroninėje mokymosi aplinkoje (Preidys, Žilinskienė, 2012), nustatant besimokančiojo mokymosi stilių (Franzoni, Assar, 2009) – parenkamas arba leidžiama pačiam besimokančiajam pasirinkti tinkamiausią mokymosi turinį arba mokymosi strategiją (Preidys, Žilinskienė, 2012), mokantis aplinkoje palaikoma adaptyvi navigacija (Weber, Brusilovsky, 2001; Paramythis, Loidl-Reisinger, 2004). Mokymosi sėkmė priklauso nuo to, kaip maksimaliai pasiekiami mokymosi tikslai – įgyjamos reikiamos dalykinės kompetencijos (žinios bei įgūdžiai, kurie yra aprašomi mokymosi rezultatais), bei kokias emocijas mokydamiesi patiria besimokantieji. Mokymosi sėkmę didžia dalimi lemia mokymosi efektyvumas, kuris priklauso nuo: noro mokytis ir mokėjimo mokytis. Pastaruosius lemia ir individualios besimokančiojo savybės, kurios skiriasi priklausomai nuo mokymosi stiliaus. Mokslininkų darbuose dažniausiai nagrinėjamas mokymosi adaptavimas atsižvelgiant į mokymosi stilius pagal *Kolb*, *Honey - Mumford*, *Felder - Silverman* klasifikacijas (Hawk, Shah, 2007). Tuo tarpu yra vos keli darbai, kuriuose *Herrmann*‘o stilių klasifikacija nagrinėta kompiuterių mokslų mokymosi kontekste (Kannangara, 2013), tačiau taikymų mokymosi sistemose neaptikta.

Disertaciniame darbe nagrinėjama programavimo mokomojo dalyko medžiagos adaptavimas prie besimokančiojo individualių savybių – jo mokymosi stiliaus pagal *Herrmann*‘o mokymosi stilių klasifikaciją. Tuo tikslu išnagrinėti programavimo mokymosi ypatumai, egzistuojančios programavimo mokymo aplinkos ir sistemos, adaptyviosios mokymosi sistemos, programiniai agentai ir agentinės sistemos, besimokančiųjų mokymosi stiliai ir mokymosi medžiagos elementai ir jų saugyklos bei mokymosi sistemos kūrimo modeliai. Darbe pateikiami šio darbo autorės sudaryti metodai: MO tipų reitingavimo pagal tinkamumą *Herrmann*‘o mokymosi stiliui metodas; patobulintas mokomojo dalyko programos susiejimo su dalykinės srities MO ir *Bloom*‘o taksonomijos lygmenimis metodas; programavimo mokymosi medžiagos elementų (MO) adaptavimo pagal besimokančiojo *Herrmann*‘o mokymosi stilių. Visi šie metodai kaip atskiri daugiaagentinės sistemos programiniai agentai integruoti į adaptyviosios programavimo mokymo priemonės (toliau adaptyvioji programavimo mokymosi sistema – APMS) modelį. Sukurtą modelį įvertino parinkti ekspertai – buvo įsitikinta modelio naudojimo kokybe, t. y. efektyvumu, našumu, produktyvumu, naudojamumu ir funkcionalumu adaptuojant mokomojo dalyko medžiagą besimokančiajam.

1.2. TYRIMO OBJEKTAS

Disertacinio darbo tyrimo objektas yra adaptyvioji programavimo mokymo priemonė (APMS).

1.3. DARBO TIKSLAS

Disertacinio darbo tikslas – ištyrus programavimo mokymo problematiką ir besimokančiųjų mokymosi galimybes bei poreikius sukurti adaptyviosios programavimo mokymo sistemos konceptualų modelį.

1.4. DARBO UŽDAVINIAI

Siekiant įgyvendinti pagrindinį tikslą iškelti tokie uždaviniai:

1 Atlikti programavimo mokymui naudojamų programinių sprendimų palyginamąją analizę adaptyvumo atžvilgiu.
2 Išskirti programavimo mokymui tinkamus mokomųjų objektų tipus ir juos suklasifikuoti pagal besimokančiojo mokymosi stilius.
3 Sukurti metodą, kuris leistų išrinkti programavimo mokomuosius objektus iš atitinkamų saugyklų atsižvelgiant į mokomojo dalyko programos reikalavimus ir besimokančiojo mokymosi stilių.
4 Parengti adaptyviosios programavimo mokymosi sistemos, adaptuojančios mokymosi medžiagą prie besimokančiojo atsižvelgiant į jo mokymosi stilių, konceptualų modelį.
5 Atlikti adaptyviosios programavimo mokymosi sistemos konceptualaus modelio vartojimo kokybinį vertinimą.

1.5. NAUDOTI TYRIMO IR DUOMENŲ ANALIZĖS METODAI

Rengiant disertacinį darbą buvo naudojami šie tyrimo ir duomenų analizės metodai:

- mokslinės literatūros ir tyrimų sisteminė analizė – ji naudota analizuojant programavimo mokymo ypatumus, metodus, problematiką, adaptyviųjų ir agentinių sistemų savybes;
- programavimo mokymo priemonių tyrimas – naudotas bandant išsiaiškinti programavimo mokymui naudojamų priemonių savybes adaptyvumo požiūriu;
- anketinė apklausa – naudota aiškinatis besimokančiųjų mokymosi poreikį;
- ekspertinis vertinimas – naudotas vertinant MO tinkamumą mokymosi stiliui ir sumodeliuotos sistemos naudojimo kokybę;
- modeliavimas – naudotas kuriant ir aprašant APMS modelį.

1.6. MOKSLINIS NAUJUMAS

- Parengtas programavimo mokomųjų objektų tipų ekspertinio vertinimo rezultatų naudojimo šiems tipams reitinguoti pagal *Herrmann*‘o mokymosi stilių metodas, leidžiantis individualizuoti mokymąsi;
- Pasiūlytas mokomojo dalyko programos susiejimo su dalykinės srities mokomaisiais objektais ir *Bloom*‘o taksonomijos lygmenimis metodas;
- Parengtas metodas, kuriuo programavimo mokomojo dalyko mokymosi medžiaga (mokomieji objektai) adaptuojama atsižvelgiant į besimokančiojo mokymosi stilių, remiantis *Herrmann*‘o mokymosi stilių klasifikacija.

1.7. GINAMI TEIGINIAI

- Mokomųjų objektų tipų tinkamumo besimokančiojo *Herrmann*‘o mokymosi stiliui ekspertinis vertinimas padeda šiuos objektus pritaikyti mokomojo dalyko mokymosi rezultatams pasiekti. Šis metodas leidžia susieti mokomųjų objektų tipus su besimokančiojo mokymosi stiliumi ir taip individualizuoti mokymąsi;
- Mokomųjų objektų susiejimo su mokomojo dalyko programa metodas leidžia sukurti technologiją, kuri pažymėtus mokomojo dalyko medžiagos elementus (mokomuosius

objektus) susieja su *Bloom*‘o taksonomijos lygmenimis ir tuo pačiu formuoja formalaus vertinimo užduočių svorius dalykinėms kompetencijoms įvertinti;
- Programavimo mokymosi sistema, adaptuojanti mokomuosius objektus prie besimokančiojo pagal jo *Herrmann*‘o mokymosi stilių, leidžia nukreipti mokymąsi siekti mokomojo dalyko programoje numatytų dalykinių kompetencijų bei įvertinti jų pasiekimo lygį;
- Konceptualiu modeliu aprašyta adaptyvioji programavimo mokymosi sistema yra efektyvi, naši, produktyvi ir naudotina (lanksti ir suderinama su kontekstu).

1.8. DARBO REZULTATŲ PRAKTINĖ SVARBA

- Naudojant pasiūlytą adaptyviosios programavimo mokymosi sistemos modelį galima sukurti sistemą, kuri pasižymėtų tokiais parametrais: efektyvumu, našumu, produktyvumu ir naudojamumu (lankstumu ir konteksto suderinamumu). Ši sistema galėtų būti integruojama kaip atskiras modulis į egzistuojančias ir plačiai naudojamas virtualias mokymosi aplinkas, pvz., *Moodle*.
- Pasiūlytas modelis gali būti taikomas kitose mokymosi srityse atitinkamai išskiriant tos srities mokymuisi labiausiai tinkančius mokomųjų objektų tipus ir mokomojo dalyko programos apraše įvardinant temas, dalykines kompetencijas ir *Bloom*‘o taksonomijos lygmenis.

1.9. DARBO APROBAVIMAS

Disertacijos rezultatai pateikti šešiose mokslinėse publikacijose, taip pat pristatyti keturiolikoje tarptautinių ir nacionalinių konferencijų bei doktorantų mokymuose.

1.10.DARBO APIMTIS IR STRUKTŪRA

Darbą sudaro: terminų ir santrumpų žodynas, 3 pagrindinės dalys – skyriai, išvados ir rezultatai, naudotos literatūros sąrašas ir priedai. Darbo apimtis yra 170 puslapiai. Tekste panaudoti 76 paveikslai, 28 lentelės ir 6 priedai. Rašant disertaciją buvo naudotasi 116 literatūros šaltinių.

Pirmajame skyriuje pateikiamas darbo įvadas, pristatomas darbo aktualumas, darbo tikslai ir uždaviniai, mokslinis naujumas, praktinė darbo reikšmė ir darbo aprobavimas.

Antrajame skyriuje nagrinėjama programavimo mokymo ypatumai, egzistuojančios programavimo mokymo aplinkos ir sistemos, adatyvios technologijos ir mokymosi sistemos, programiniai agentai ir agentinės sistemos, besimokančiųjų mokymosi stiliai ir mokymosi medžiagos elementai ir jų saugyklos, mokymosi sistemos kūrimo modeliai.

Trečiajame skyriuje aprašomas adaptyviosios programavimo mokymosi sistemos konceptualaus modelio projektavimas. Sistema projektuojama kaip daugiaagentinė sistema, naudojanti specializuotoje mokomųjų objektų saugykloje saugomus kokybiškus programavimo mokomojo dalyko programą atitinkančius (pažymėtus) mokomuosius objektus. Mokomieji objektai yra besimokančiajam parenkami ir reitinguotai pateikiami atsižvelgiant į jo *Herrmann*‘o mokymosi stilių, o mokomųjų objektų tipų tinkamumas mokymosi stiliui nustatomas panaudojant ekspertinio vertinimo metodą.

Ketvirtajame skyriuje pateikiamas sukurto adaptyviosios programavimo mokymosi sistemos modelio vartojimo kokybės vertinimas. Naudojamas ekspertinio

vertinimo metodas, kuris leidžia įvertinti sukurto sistemos modelio naudojimo kokybę tokiais požiūriais: efektyvumo, našumo, produktyvumo ir naudojamumo (lankstumo ir konteksto suderinamumo).

Darbo pabaigoje pateikiamas rezultatų apibendrinimas ir išvados.

BENDROSIOS IŠVADOS IR REZULTATAI

1. Išanalizavus programavimo mokymo problematiką, programavimo mokymo ir adaptyviąsias sistemas, besimokančiųjų mokymosi stilių įtaką mokymosi rezultatams, nustatyta, kad:
   1.1. Svarbus mokymosi individualizavimas, nes besimokančiajam programuoti yra būdingas savitas informacijos apdorojimo stilius, į kurį atsižvelgiant būtų padidinamas mokymosi efektyvumas. Kaip parodė žvalgomasis tyrimas, mokymosi individualizavimo poreikį akcentuoja ir patys besimokantieji.
   1.2. Nėra adaptyviųjų programavimo mokymosi sistemų, kurios automatiškai kurtų mokymosi kursą ir jį adaptuotų prie besimokančiojo individualių savybių atsižvelgiant į jo mokymosi stilių. Tikslinga sukurti sistemą, kuri būtų universali ir automatizuotų didžiąją dalį kurso kūrėjo rutininių darbų, atsižvelgdama į pateikiamą mokomojo dalyko apraše pateikiamą informaciją. Naudojant programavimo mokomuosius objektus ir automatinį programų žymėjimą gali būti kuriamas mokomojo dalyko apraše nurodytą turinį atitinkantis mokymosi kursas. Veiksmams ir reakcijoms į aplinkos faktorius automatizuoti sistemoje tikslinga naudoti programinius agentus.
   1.3. Nagrinėtuose mokslo darbuose nebuvo išskirti ir aprašyti programavimo mokymuisi tinkami mokomųjų objektų tipai ir jie susieti su mokymosi stiliais pagal *Herrmann*‘o mokymosi stilių klasifikaciją. Šis susiejimas užtikrina mokymosi individualizavimą, kai atsižvelgiant į individualias besimokančiojo savybes mokymosi metu adaptuojama mokymosi medžiagą. Mokymosi kurse naudojamų programavimo mokomųjų objektų tipų parinkimą įtakoja analizės metu išskirtos mokymosi veiklos: informacijos šaltinių (sąvokų, terminų, konceptų) studijavimas; pavyzdžių studijavimas; grafinės informacijos studijavimas; vaizdo medžiagos studijavimas; modeliavimas bei imitavimas; užduočių sprendimas; problemų sprendimas; projektų rengimas.
   1.4. Projektuojamoje adaptyviojoje programavimo mokymosi sistemoje naudojamam dalykinių kompetencijų vertinimui, atsižvelgiant į *Bloom*‘o taksonomijos lygmenis, užtikrinti turi būti atliktas mokomojo dalyko programos (temų ir dalykinių kompetencijų) tikslus susiejimas su mokomaisiais objektais ir *Bloom*‘o taksonomijos lygmenimis.
2. Remiantis analizės metu nustatytais programavimo mokymo ypatumais, besimokančiųjų individualiomis savybėmis ir adaptyviųjų bei mokymosi sistemų kūrimo metodais sukurtas adaptyviosios programavimo mokymosi sistemos modelis, kuris:
   2.1. Susieja *Herrmann*‘o mokymosi stilius su mokomųjų objektų tipais. Parengtas mokomųjų objektų tipų ekspertinio vertinimo metodas leidžia geriau adaptuoti mokomuosius objektus prie besimokančiojo mokymosi stiliaus;
   2.2. Suteikia galimybę įvertinti mokomųjų objektų kokybę prieš siūlant juos mokymosi kurse. Mokomųjų objektų kokybinio vertinimo metodo integravimas į adaptyviąją programavimo mokymosi sistemą leidžia užtikrinti mokymosi

medžiagos kokybę, nes specializuotoje mokomųjų objektų metaduomenų saugykloje (SMOS) saugomi tik kokybiški mokomieji objektai;

2.3. Leidžia susieti mokomuosius objektus su programavimo mokomojo dalyko temomis, dalykinėmis kompetencijomis ir *Bloom*‘o taksonomijos lygmenimis, kas sudaro galimybę įvertinti dalykinių kompetencijų pasiekimo lygį atsižvelgiant į iš anksto apibrėžtas taisykles;

2.4. Leidžia individualizuoti mokymąsi, t. y. rekomenduoti besimokančiajam jo individualias savybes (mokymosi stilių) labiausiai atitinkantį mokymosi kursą, parenkant jo mokymosi stilių atitinkančių mokomųjų objektų tipų eilę.

3. Atlikus adaptyvios programavimo mokymosi sistemos konceptualaus modelio ekspertinį vertinimą nustatyta, kad:
   - Modelyje siūlomi metodai leidžia besimokančiajam pateikti tinkamus mokomuosius objektus, tinkančius siekti dalyko programoje numatytų dalykinių kompetencijų, tad galima teigti, kad sistema yra efektyvi;
   - Modelyje siūlomi metodai leidžia taupyti laiką renkantis tinkamus konkrečios užduotims atlikti mokomuosius objektus, todėl sistema yra naši;
   - Modelyje siūlomi metodai leidžia vartotojui padidinti mokomųjų objektų kiekį, todėl metodai yra produktyvūs;
   - Siūlomi metodai leidžia adaptuoti mokomojo dalyko medžiagą prie besimokančiojo individualių savybių, todėl sistema yra lanksti;
   - Modelyje siūlomi metodai gali būti naudojami įvairiuose mokymosi kontekstuose, šitaip užtikrinamas konteksto suderinamumas.
4. Naudojant pasiūlytą adaptyviosios programavimo mokymosi sistemos konceptualų modelį galima sukurti sistemą, adaptuojančią mokomojo dalyko medžiagą prie besimokančiojo ir pasižyminčią efektyvumu, našumu, produktyvumu, naudojamumu (lankstumu ir konteksto suderinamumu).
5. Naudojant pasiūlytą adaptyviosios programavimo mokymosi sistemos konceptualų modelį galima sukurti sistemą, kuri pasižymėtų tokiais parametrais: efektyvumu, našumu, produktyvumu ir naudojamumu (lankstumu ir konteksto suderinamumu). Ši sistema galėtų būti integruojama kaip atskiras modulis į egzistuojančias ir plačiai naudojamas virtualias mokymosi aplinkas, pvz., *Moodle*.
6. Pasiūlytas modelis gali būti taikomas kitose mokymosi srityse atitinkamai išskiriant tos srities mokymuisi labiausiai tinkančius mokomųjų objektų tipus ir mokomojo dalyko programos apraše įvardinant temas, dalykines kompetencijas ir *Bloom*‘o taksonomijos lygmenis.

TRUMPOS ŽINIOS APIE AUTORĘ

Jūratė Urbonienė gimė 1974 m. liepos 30 d. Šiaulių mieste.
1992 m. baigė Švenčionių Zigmo Zemaičio vidurinę mokyklą. Matematikos mokslų bakalauro ir vidurinės mokyklos matematikos ir informatikos mokytojo kvalifikaciją 1996 m. įgijo Vilniaus pedagoginiame universitete (dabar Lietuvos edukolegijos universitetas) Matematikos fakultete. Tame pačiame universitete 1999 m. įgijo matematikos mokslų (informatikos) magistro laipsnį ir gimnazijos informatikos mokytojo kvalifikaciją. 2007 m. Kauno Technologijos universitete įgijo informatikos inžinerijos magistro kvalifikacinį laipsnį.
2009–2013 m. studijavo Vilniaus universiteto Matematikos ir informatikos instituto doktorantūroje (technologijos mokslai, informatikos inžinerija).
2010, 2011 ir 2012 m. buvo priimta į doktorantų vasaros mokyklą ir gavo finansavimą.
Nuo 1998 m. Jūratė Urbonienė dirba Utenos kolegijoje (iki 2000 m. buvusioje Utenos aukštesniojoje verslo mokykloje) dėstytoja, o nuo 2009 m. buvo paskirta vadovauti Technologijų (dabar Elektros ir informatikos inžinerijos) katedrai, 2011 m. atestuota docento pareigoms.
El. pašto adresas: jurate99@gmail.com

JŪRATĖ URBONIENĖ
DESIGN OF ADAPTIVE PROGRAMMING TEACHING TOOLS
Summary of Doctoral Dissertation
Technological sciences, informatics engineering (07 T)

JŪRATĖ URBONIENĖ
ADAPTYVIŲJŲ PROGRAMAVIMO MOKYMO PRIEMONIŲ PROJEKTAVIMAS
Daktaro disertacijos santrauka
Technologijos mokslai, informatikos inžinerija (07 T)